Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2008-035 August 11, 2006

Matching Sets of Features for Efficient Retrieval and Recognition

Kristen Lorraine Grauman

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Matching Sets of Features for Efficient Retrieval and Recognition

by

Kristen Lorraine Grauman

Bachelor of Arts, Computer Science
Boston College, 2001

Master of Science, Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2003

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2006

Author .....
Department of Electrical Engineering and Computer Science
August 11, 2006

Certified by .....
Trevor Darrell
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Matching Sets of Features for Efficient Retrieval and Recognition

by
Kristen Lorraine Grauman

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

In numerous domains it is useful to represent a single example by the collection of local features or parts that comprise it. In computer vision in particular, local image features are a powerful way to describe images of objects and scenes. Their stability under variable image conditions is critical for success in a wide range of recognition and retrieval applications. However, many conventional similarity measures and machine learning algorithms assume vector inputs. Comparing and learning from images represented by sets of local features is therefore challenging, since each set may vary in cardinality and its elements lack a meaningful ordering.

In this thesis I present computationally efficient techniques to handle comparisons, learning, and indexing with examples represented by sets of features. The primary goal of this research is to design and demonstrate algorithms that can effectively accommodate this useful representation in a way that scales with both the representation size as well as the number of images available for indexing or learning.

I introduce the *pyramid match* algorithm, which efficiently forms an implicit partial matching between two sets of feature vectors. The matching has a linear time complexity, naturally forms a Mercer kernel, and is robust to clutter or outlier features, a critical advantage for handling images with variable backgrounds, occlusions, and viewpoint changes. I provide bounds on the expected error relative to the optimal partial matching. For very large databases, even extremely efficient pairwise comparisons may not offer adequately responsive query times. I show how to perform sub-linear time retrievals under the matching measure with randomized hashing techniques, even when input sets have varying numbers of features.

My results are focused on several important vision tasks, including applications to content-based image retrieval, discriminative classification for object recognition, kernel regression, and unsupervised learning of categories. I show how the dramatic increase in performance enables accurate and flexible image comparisons to be made on large-scale data sets, and removes the need to artificially limit the number of local descriptions used per image when learning visual categories.

Thesis Supervisor: Trevor Darrell
Title: Associate Professor

Acknowledgments

I am fortunate to have had remarkable mentors, colleagues, family, and friends who have offered me invaluable guidance and support, both prior to and during my time as a graduate student.

I would like to thank Trevor Darrell for being an excellent advisor, mentor, and collaborator while I have been at MIT. I am grateful not only for his contributions to the ideas in this research, but also for his general upbeat attitude, flexibility, and friendship, which have all enhanced my graduate career significantly. Trevor has the fine ability to balance giving me independence to pursue my ideas with offering insights and a fresh perspective at critical moments. He has been a reliable source of advice on matters large or small, and I think his genuine dedication to the group and our research is exceptional.

Leslie Pack Kaelbling, Bill Freeman, and Pietro Perona have been great thesis committee members, and their careful reading of this thesis and thoughtful feedback have strengthened it. I am grateful to Piotr Indyk as well: his work inspired me to explore this topic area, and he was always willing to discuss it with me.

My fellow students and members of the Vision Interface group have contributed a great deal to my experience as a graduate student. I am particularly thankful to have had the opportunity to collaborate and share an office with Greg Shakhnarovich, whose sense of humor is always entertaining, and whose pure enthusiasm about computer vision I admire. It has been great to work with John Lee over the last year; he is very talented, and I have enjoyed our discussions. I would like to thank all the other members of the group for being friendly and helpful colleagues, especially Mario Christoudias, Neal Checka, David Demirdjian, Louis-Philippe Morency, Ariadna Quattoni, Sybor Wang, and Tom Yeh.

I am very happy to have had the opportunity to be a graduate student at MIT. There is such a unique combination of openness, love for learning, and just plain quirkiness here that makes it an incredible place to study and do research. There is no one person to thank in this regard, but I am grateful to the faculty and student body as a whole for fostering this special culture. I am also thankful for the Department of Energy Computational Science Graduate Fellowship that funded me; it provided me an even greater sense of research freedom, particularly when I was a new graduate student.

As an undergraduate, I benefitted greatly from the guidance and excellent teaching abilities of my computer science professors at Boston College, especially Professors William Ames, Margrit Betke, James Gips, Peter Kugel, Robert Muller, Edward Sciore, and Howard Straubing. They collectively sparked and then developed my interest in computer science, and I sincerely appreciate and admire their dedication to students. Margrit's course on computer vision inspired me to consider vision as a research area, and as my undergraduate advisor she helped me begin to understand the process of research itself. I am very grateful for her direction and encouragement.

Finally, I want to thank my friends and family who have constantly supported and inspired me, though I hope that they already know my appreciation much more completely than these few words in an acknowledgement section can express. My dear friend Ari Shapiro

has a big smile that is matched by his equally big heart; I couldn't ask for a more fun and compassionate friend. Thank you Ari. I'm lucky to have a one-of-a-kind brother Greg and wonderful sister-in-law Belinda. I have lifelong role models in my aunts Susie, Ellen, Carol, Julie, and Nancy; thanks for giving me plenty of good reasons to look up to you. I love and admire my grandparents Angeline and George Slezak, and my late grandparents, Philip and Lorraine Grauman, and want to thank them for always nurturing and encouraging me.

I am deeply grateful to my parents, Robert and Karen Grauman, who are incredibly wise and generous. Mom and Dad, completing this degree (among many other things) would not have been possible without your love and support.

Finally, my heartfelt appreciation goes to Mark Stephenson, who makes every day better.

Contents

List of Figures

Chapter 1

Introduction

1.1 Recognizing and Retrieving Images

The visual world is a rich and complex source of information. We as humans constantly draw on it, whether for purely practical purposes—such as navigation, recognition of familiar people and objects, or manipulating tools and devices, or for purposes that are more intricate and subtle—such as inferring the function of an unfamiliar object, interpreting gestures and facial expressions, or detecting abnormal activity. In computer vision research, we want to develop the algorithms and representations that will allow a computer to autonomously analyze visual information, with the belief that in doing so we will see a significant impact on our productivity and understanding of the world around us.

The fundamental problem I address in this dissertation is how to efficiently match and index sets of features. This problem underlies many computer vision tasks, including object recognition and image retrieval. In this chapter I will first discuss the primary challenges of recognition and retrieval at a high level, and then I will overview the contributions of my research.

Recognition is the core problem of learning visual categories and then identifying new instances of those categories. It is generally considered by vision researchers as having two types: the specific case and the category case. In the specific case, we seek to identify instances of a particular object, place, or person—for example, Charlie Chaplin's face, the Taj Mahal, or a certain magazine cover. In contrast, at the category level, we seek to recognize different instances of a generic category as belonging to the same conceptual class—for example, butterflies, coffee mugs, or umbrellas. At the category level we can expect that there may be large variations in appearance among different instances of the same class.

In content-based image retrieval, the goal is to use the visual content of an image query to identify the most relevant images within a database, under some defined notion of similarity. In contrast to keyword-based indexing and retrieval, where images are organized according to simple textual tags, content-based image retrieval aims to sort images directly using their visual information. Retrieval and recognition are strongly related in that in both we must

(a) Illumination (b) Object pose (c) Clutter

(d) Occlusion (e) Intra-class appearance (f) Viewpoint

Figure 1-1: To robustly accommodate real-world images, recognition algorithms must face an extraordinary amount of variation, as depicted by these strikingly different images produced by instances of the same object category.

understand what makes two images visually (or even semantically) similar; however, the retrieval problem also implies that a potentially very large amount of data must somehow be indexed efficiently in order to perform real-time searches.

The recognition and retrieval problems are especially compelling now due to the numerous, rapidly growing sources of visual data that are available, including personal digital photo libraries, streaming web cameras, satellite imagery, cell phone cameras, medical imaging devices, surveillance videos, commercial product catalogs, and archival databases. Today it is increasingly inexpensive to capture and store image and video data, yet automatically organizing and indexing that data presents significant technical and computational challenges.

The flexibility demanded from recognition and retrieval algorithms that are expected to operate in real-world conditions is staggering. While the human visual system itself is not fully understood, it is clear that humans are capable of recognizing objects with remarkable speed and versatility. We can not only identify an object under a wide range of illuminations, against different backgrounds, or in a variety of poses, but we can also take relatively few instances of an object class and generalize to recognize novel instances containing significant differences in appearance. For a computer to have this versatility, however, recognition algorithms must take into account that even moderate changes in an object's pose, illumination, or background may yield drastic changes in the 2-D array of

pixels that is processed, as illustrated by the koala images in Figure 1-1.

However, robustness to these considerable modes of variation must not be achieved at the expense of computational feasibility. As in most computer science disciplines, vision researchers are often caught in the tension between descriptiveness and tractability. Consider the following indicators of the high-dimensional input and parameter spaces that naturally occur in recognition problems, as well as the sheer volume of image and video data we would like to index and organize; there are:

- Thousands to millions of pixels in an image
- An estimated 3,000-30,000 human recognizable object categories [11]
- Over 30 degrees of freedom in the pose of articulated objects (humans)
- Billions of images indexed by Google^{TM} Image Search
- Over 18 billion prints produced from digital camera images in 2004
- 295.5 million camera phones sold in 2005

There is a real struggle: on the one hand, we have a desire to build increasingly flexible models for recognition and retrieval, while on the other hand, we have the functional need to make the models and inference processes practical to compute.

1.2 Thesis Overview

I believe that to be successful, recognition and retrieval algorithms must not only capture robustness to widely varying image conditions, but they must do so efficiently. My research embraces these dual goals: I have created a method that enables efficient learning from *sets of features*—a prevalent and powerful image representation—and have developed a technique for fast content-based search over large image databases.

1.2.1 Sets of Local Features

Earlier work in the field focused on global representations that describe each image with a *single* vector of attributes, such as a color histogram or a list of filter responses. While vector representations do permit the direct application of standard machine learning techniques and distance functions, they are known to be problematically sensitive to realistic image conditions. Much recent work shows that decomposing an image into its component parts (or *local features*) grants resilience to image transformations and variations in object appearance. Typically, an *interest operator* is used to identify numerous salient regions in an image; then, for each detected region, a (vector) feature descriptor is formed (see Figures 1-2 and 1-3). The result is one set of local appearance or shape description vectors per image.

Strong similarity between multiple local portions of related images may often be discovered even when the images' global appearance happens to obscure this likeness. Local features

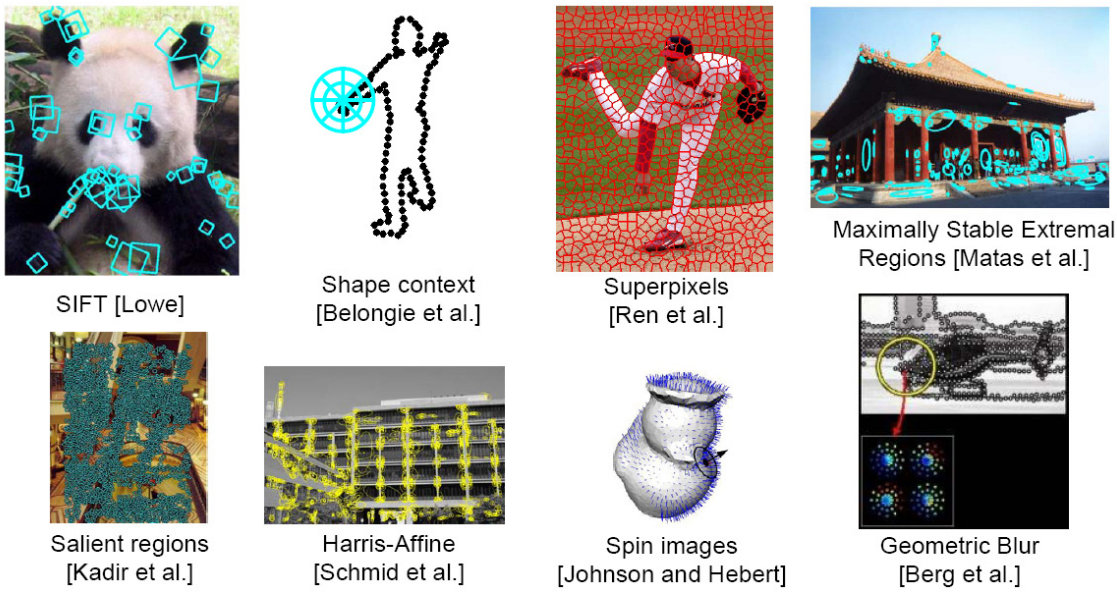


Figure 1-2: Researchers have introduced various ways to extract local features from images, many of which include explicit invariance properties for common geometric and photometric transformations. Details on each feature type are provided in Chapter 2.

provide a more flexible representation and can often be more reliably matched for several reasons:

- **Change globally, look locally:** An image's global representation can change considerably due to variations in viewpoint, object pose, or deformations; however, changes within any given *local* region will be smooth and less dramatic, simply due to the smaller amount of viewable area.
- **Isolate occlusions:** An object may be severely occluded by another object. A global representation will suffer proportionally, but for local representations, any local parts that are still visible will have their descriptions remain intact. (For instance, the global color histogram for an image of a panda occluded by a red balloon will not appear similar to the color histogram for an image with a panda alone; however, some of the histograms taken from local "blob-like" regions in either image will still appear very similar.)
- **Isolate clutter and the background:** Similarly, while the global description may be overwhelmed by large amounts of background or clutter, small parts of an image containing an actual object of interest can emerge if we describe them independently by regions.
- **Accommodate partial appearance variation:** When instances of a category can vary widely in some dimensions of their appearance, their commonality may be best captured by a part-wise description. (For example, consider different models of aircraft: they may exhibit significant differences in global shape, but still share local structures such as windows or landing gear wheels.)
- **Invariant local descriptors:** Researchers have developed local descriptors that

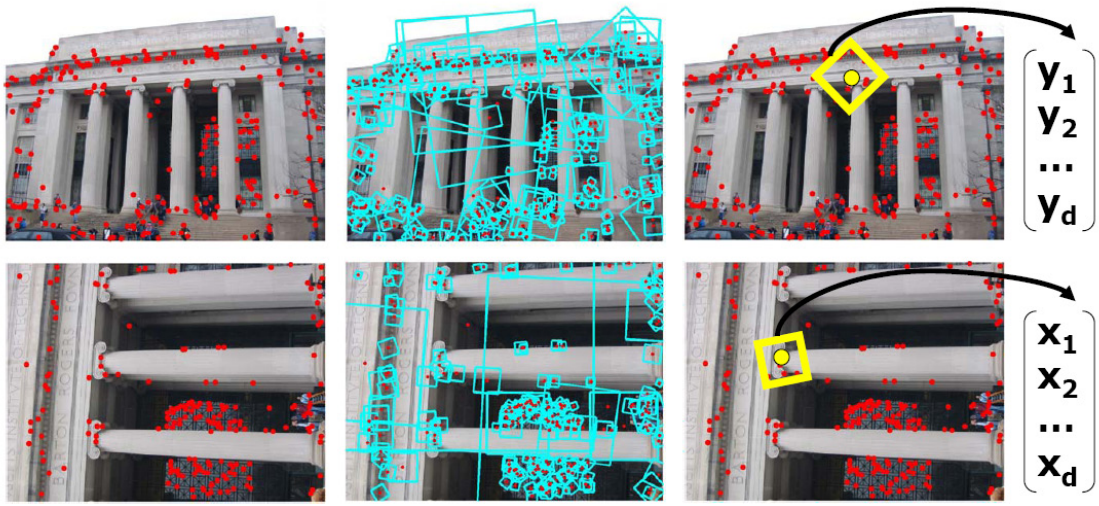


Figure 1-3: While two images of the same scene may globally be quite different, strong similarity may be present in their sets of local features. Interest operators identify salient points within an image, and they often select points that will be repeatedly detected under varying scales (left column). A local patch or region around each detected point is extracted from the image and used to form a descriptor (center column). Descriptors designed to have certain invariance properties will produce very similar feature vectors for the same local part of an object, even if it is viewed at a different orientation, scale, or translation (right column). Here, the building is viewed at two distinct scales and orientations, but the invariant local feature descriptors $\mathbf{x}, \mathbf{y} \in \Re^d$ computed from the same patch on the building (indicated by a yellow box in each image) will still have similar values in both instances.

are explicitly invariant to common transformations, such as illumination changes, rotations, translations, scaling, or all affine transformations (see Figure 1-3).

1.2.2 Contributions

The problem, however, is that this appealing representation—a set of vectors—renders traditional distances and learning algorithms unsuitable. Many conventional similarity measures and machine learning techniques assume (uniform-length) vector inputs, but with local representations, each image produces a variable number of features, and there is no ordering among features in a single set.

My work provides an important step towards solving this problem. In this research I present computationally efficient techniques to handle comparisons, learning, and indexing with sets of local features. The primary goal of this dissertation is to provide algorithms that can effectively accommodate this useful representation in a way that scales with both the set size as well as the number of images to be indexed or learned from. I treat the core of image retrieval and recognition as a matching problem: if two objects or images are similar, there should be a strong correspondence between their parts' appearance.

I develop an efficient method called the *pyramid match*, which forms a matching between

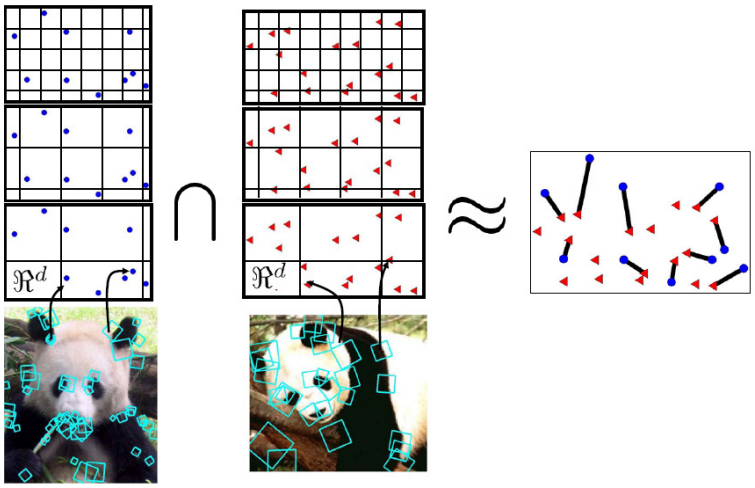


Figure 1-4: The pyramid match measures the partial matching similarity between sets of vectors. The approximation relies on a multi-resolution partitioning of the local descriptor feature space to efficiently form correspondences between points in two sets via weighted intersections.

two sets of feature vectors, and show how it can be effectively used as measure of similarity between images represented by unordered, variable-sized sets of local features. The cost measured by the pyramid match approximates the cost measured by the optimal correspondences between feature sets of unequal cardinality (i.e., the *partial matching* that optimally maps points in the lower cardinality set to some subset of the points in the larger set, such that the summed cost between matched points is minimal). While an optimal partial match would cost time cubic in the number of features per input set, my approximation requires only *linear* time.

The basic idea of the pyramid match is to use a multi-resolution partitioning of the feature space together with a weighted intersection measure to directly count off matches formed at different distance ranges between the points in two sets (see Figure 1-4). The intuition is to start collecting groups of matched points from the bottom of the pyramid up, i.e., from within increasingly larger partitions. First we consider matching the closest points that fall together within small bins, and as we proceed to the coarser resolutions in the pyramid we allow increasingly further points to be matched. The key is that the matching cost can be measured via these histogram pyramids without even computing pairwise distances between the vectors in two point sets.

I provide theoretical bounds on the expected error for the pyramid match relative to the optimal partial matching, and I also describe how the matching approximation can benefit from knowledge of the distribution of points in the feature space, when such knowledge is available. By exploiting the features' underlying structure, the pyramid match can achieve accurate and efficient results even in high-dimensional feature spaces.

The matching approximation may be used directly as a distance or similarity measure, which is an important tool for many vision tasks. For example, image retrieval requires identifying the nearest examples to a given query; clustering image descriptions allows automatic

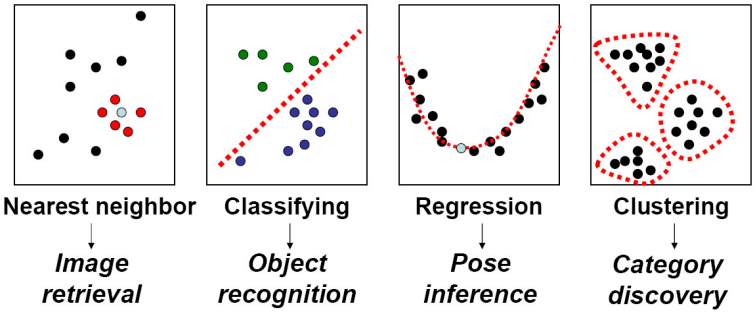


Figure 1-5: A similarity measure between image representations is central to several vision tasks.

database organization and is the foundation of unsupervised learning from examples; it is possible to learn decision boundaries or functions for a parameter of interest if we know how all examples are situated relative to one another in some space (see Figure 1-5).

For very large databases where it is infeasible to index with a naive linear scan of all items, I show how to perform sub-linear time retrievals with local feature representations and randomized hashing techniques. This capability lays the groundwork for modern data-rich approaches to computer vision.

The matching also serves as a conduit to kernel-based learning methods for set inputs. I show that the pyramid match naturally forms a positive-definite function, making it valid for use in kernel methods whose optimal solutions and convergence guarantees require Mercer kernels. This makes it possible to access numerous existing machine learning methods with the set representation, including techniques for discriminative classification, regression, dimensionality reduction, and clustering.

The efficient matching and retrieval framework I have developed has a very simple, general architecture that may be extrapolated to other problems within vision and beyond. My results are concentrated on vision problems in recognition and retrieval, but establishing the correspondence between sets of image features is an essential problem underlying several other vision objectives, from solving for the geometry of a scene from multiple views, to localizing objects composed of parts, to estimating the motion of an articulated object.

In addition, the set-of-vectors representation also arises in domains outside of vision, as it is often natural and meaningful to represent a single example by the collection of local features or parts that comprise it. For instance, in natural language processing, documents or topics may be represented by sets (bags) of word features; in computational biology, a disease may be characterized by sets of gene-expression data from multiple patients. In this dissertation I present the algorithmic details first, followed by the applications, so that a reader interested only in alternative application areas to vision may choose to focus on the chapters about the core methods, Chapters 3 through 5.

1.3 Road Map

In the following chapter I give a more detailed overview of the problem setting and a discussion of related work. Then in Chapter 3 I define the core matching algorithm. In Chapter 4 I describe an extension to the matching that leverages structured feature distributions within the approximation. Chapter 5 discusses randomized hashing techniques for sub-linear time search of large databases on the basis of matching similarity. Chapter 6 empirically evaluates the quality of the matching approximation relative to the optimal measure.

Once the majority of the algorithmic components have been presented in these early chapters, I then provide experiments and results demonstrating these ideas applied to several computer vision problems. Specifically, in Chapter 7 I present image and shape retrieval results with a variety of data sets and demonstrate the impact of hashing for approximate nearest-neighbor search with very large image databases. Following that, I show how the pyramid match may be employed as a kernel for structured inputs in Chapter 8, and describe cases of supervised learning where it is very effective, namely discriminative categorization of objects and inference of human body pose. I present one non-vision application in this chapter as well, demonstrating inference of publication time from local feature representations of documents. Finally, in Chapter 9 I give a method for organizing a collection of unlabeled images into categories based on their sets of partially matching features. Throughout, I support each class of results with direct comparisons against state-of-the-art methods and take advantage of several benchmark data sets.

Chapter 2

Background and Related Work

In this chapter I frame the problem setting for my research and overview related work. I first describe the local feature image representation and its advantages, and then I discuss approaches vision researchers have considered for working with this representation. I organize my discussion of previous work in vision and machine learning into two groups: approaches for measuring correspondence-based similarity from the image indexing literature (Section 2.2), and approaches for (primarily kernel-based) learning from sets of local features (Section 2.3). Following this coverage of the relevant prior work in vision and learning, in Section 2.4 I review related techniques for approximate matching from the theory of computation literature. Throughout I draw comparisons and contrasts with my own work.

2.1 Image Matching and Image Representations

Image matching and recognition are important computer vision problems with a variety of applications, such as content-based image retrieval, object and scene categorization, and video data mining. The task of identifying similar objects and scenes remains challenging due to viewpoint or lighting changes, deformations, and partial occlusions that may exist across different examples. The very same object can produce dramatically different images depending on its pose relative to the camera, the current illumination, or simply the background that it is photographed against. On top of these complicating factors, different instances of objects from the same category can exhibit significant variations in appearance.

To compute anything with an image, a choice of representation must be made. How should an image be described to capture the relevant visual cues it contains? Earlier work in the field focused on global representations that describe each image with a single vector of attributes. Examples of global representations include a color or grayscale histogram, a list of responses to a set of filter banks, or simply a list of intensities obtained by stacking the columns of an image into a single vector. While vector representations do permit the direct application of standard machine learning techniques and distance functions, they are known to be prohibitively sensitive to realistic image conditions, and often cannot give adequate descriptions of an image's local structures and discriminating features.

A global representation that assumes an ordering can be overly constraining. For example, if a face class is represented by ordered lists of intensities taken from examples of frontal faces, when a profile face is encountered, the nose pixels will no longer be in the "right" position, and this global representation will fail. A global representation also typically makes the assumption that all parts of an image should influence the description equally.[1] This assumption often does not hold; if an object of interest is posed against a large background, that background could overwhelm the global description, making it difficult or impossible to identify strong similarity between that object and another view of it against a different background.

Much recent work shows that decomposing an image into its component parts (or "local features") grants resilience to image transformations and variations in object appearance. Typically, an interest operator is used to identify numerous salient regions in an image; then, for each detected region, a (vector) feature descriptor is formed. The descriptors are often designed to be invariant to transformations likely to occur across images, such as photometric changes or various geometric transformations, including rotations, translations, scaling, or all affine transformations.

Lowe has developed a widely-used local descriptor based on histograms of local image gradients called Scale Invariant Feature Transform (SIFT) [73]. To identify scale- and orientation-invariant interest points, Lowe uses a difference-of-Gaussian function. The SIFT descriptor has been shown to be robust under shape distortion and illumination changes, and it has also been shown to be very resilient in an independent study [78]. A low-dimensional version of SIFT is designed in [63]. I make use of the SIFT descriptor in various experiments in this work to describe the appearance of local regions.

Belongie et al. design a local descriptor called the shape context histogram to represent shape as it appears along points on an edge or contour [7]. In a related approach, Berg and Malik develop the geometric blur descriptor in [10] to perform template matching that is resistant to affine distortions, and show its effectiveness for object recognition in [9]. Johnson and Hebert introduce a local surface descriptor for 3-D shapes called the "spin image" in [59]. The "superpixels" of Ren and Malik are small local regions of homogenous texture or color found with a low-level segmentation computed with Normalized Cuts [90]. The Maximally Stable Extremal Region detector of Matas and colleagues locates blob-like affine-invariant regions that have consistent intensity within them [76]. The Harris-Affine detector of Mikolajczyk and Schmid detects points that are invariant to scale and affine transformations [79]. Kadir and Brady create a hierarchy of salient regions that operate across feature space and scale [61]. Biologically-inspired features and learning paradigms have been explored by Serre and Wolf and colleagues [94, 114].

In the recognition literature, many researchers have turned to these local feature representations that describe the appearance or shape within confined neighborhoods in the image, including [73, 79, 7, 34, 9, 44, 53, 31, 108], whose work I describe individually below. I use existing techniques to detect and describe the local features themselves and do not contribute new ideas on the representation side of the problem. Given a suitable local feature description for a given task, my research focuses on how to compare, index, and recognize

[1] However, see work by Shakhnarovich for employing category- or task-specific feature selection on a global representation [96].

with local representations.

As discussed in the previous chapter, it is important for many tasks to be able to compare any two image descriptions. With global descriptions, every image is a vector, and in that case every point in the plots in Figure 1-5 denotes one of these (high-dimensional) vectors. In this case, standard vector-based distances and machine learning algorithms are applicable.

The problem, however, is that the appealing local representations do not fit this picture, and for this reason they render many traditional distances and learning algorithms inapplicable. Each image may produce a variable number of local features, and there is no ordering among features in a single set. This means it is not possible to apply standard vector-based algorithms to compare and learn from sets of local features.

In general, existing computer vision and machine learning methods compare feature sets by searching for explicit correspondences between their elements (which is too costly for large scale problems), fitting parametric distributions to the sets (which makes possibly restrictive assumptions about the data), or comparing histograms over a set of prototype features (which must be found by clustering).

In the following section I review techniques for image indexing that measure the similarity between sets of local features, and then in Section 2.3 I describe machine learning techniques for handling the set representation.

2.2 Measuring Similarity Between Sets of Features

How can examples having a variable-sized, unordered set representation best be compared? Clearly, general-purpose distance or similarity measures that operate on vectors of a fixed number of attributes (e.g., Euclidean norms, correlation coefficient, dot product) cannot be directly applied. Current image matching techniques that do handle sets of features typically judge similarity via a voting scheme [73, 77, 105, 95] or by comparing histograms measuring the frequency of occurrence of a set of prototype features [102, 26, 68]. A number of methods also compare sets based on the explicit one-to-one correspondences found between features [20, 7, 92, 40, 9]. Below I describe each of these types of approaches in turn, along with their limitations.

2.2.1 Voting

A number of recent matching techniques extract the local features having strong invariance properties in all images, and then use voting to rank the database images in similarity: the query image's features vote independently for features from the database images (where votes go to the most similar feature under some distance, e.g., L_2), possibly followed by a verification step to account for geometric relationships between the features. In the method of Mikolajczyk and Schmid, each scale-invariant interest point in a query image votes for images in the database containing an interest point within a thresholded distance from

itself [77]. Similarly, Tuytelaars and Van Gool use affine moment invariants to cast votes for similar images in the database [105]. The method developed by Shaffalitzky and Zisserman for matching scenes first uses voting to identify candidate matches, then applies a series of steps to verify geometric consistency within larger neighborhoods [95]. Lowe shows how an object's keypoints can be matched independently via a thresholded approximate similarity search to all of the keypoints extracted from the database images; clusters of three matches that agree in pose indicate the object's presence in a database image [73].

When sufficiently salient features are present in an image, methods based on the voting scheme may successfully identify good matches in the database, particularly when matching multiple images of the same object or scene [42]. In fact, the local features designed to be invariant to various transformations were initially intended to be used for stereo or multi-view matching of images containing the same scene, and only recently have they begun to be explored for more general purpose category-level matching.

However, using a query image's features to independently index into the database ignores possibly useful information that is inherent in the co-occurrence of a set of distinctive features; voting implies non-injective correspondences. It is also not possible with this approach to distinguish between instances where an object has varying numbers of similar features. Thus, the voting-based methods typically filter through the initial feature matchings and follow up with geometric verification steps. A naive exhaustive search for the most similar features would make the voting technique computationally prohibitive for significant amounts of data. In practice, an approximate similarity search technique must be used, such as the Best-Bin-First search technique employed in [73].

My work differs from the voting-based techniques [105, 77, 73, 95] in that I advocate matching sets of features; rather than match features within an image independently, my method considers the joint statistics of the local features as a whole when matching. As I show in Chapter 7, taking these joint statistics into account can be critical to performance for matching problems involving images from generic object categories or textures.

2.2.2 Bags of Prototypical Features

Other matching approaches have taken feature co-occurrences into account by using vector quantization to represent an image by its frequency counts of prototypical feature occurrences [70, 102, 26, 67, 68]. Sivic and Zisserman suggested the application of text retrieval techniques to image matching in [102]; vector quantization (VQ) is applied to affine-invariant regions collected from video data, and each image is represented by a fixed-length vector of weighted frequencies of the pre-established feature prototypes. Then, images in the database are ranked in similarity to a user-segmented query region based on their frequency vectors' normalized scalar product.

Csurka and colleagues also apply VQ to affine-invariant appearance features, and then train a classifier with each image represented by its frequencies of prototypes [26]. More recently, Nister and Stewenius have shown that a hierarchical quantization of image features provides a scalable means of indexing into a very large feature vocabulary [85]. However, in their work the goal is not correspondence-based set matching, but rather indexing individual

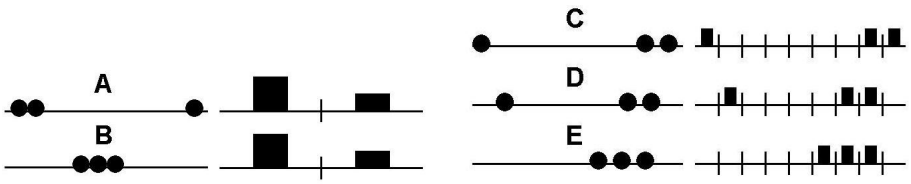


Figure 2-1: Comparing quantized feature sets with a bin-by-bin similarity measure is sensitive to bin size. If bins are too wide, discriminative ability is lost (Point sets A and B produce the same prototype frequencies in spite of their distinct distributions). If bins are too narrow, features that are very similar are placed in separate bins where they cannot be matched (D and E are considered closer than C and D, which are perceptually more similarly distributed). Cross-bin measures avoid the bin sensitivity issue, since features are matched based on their similarity to one another, not their assigned bin placement.

feature vectors into a database, meaning the distribution of features within an image is not incorporated in the matching.

In a method developed by Lazebnik and colleagues [67], textures are recognized based on their histogram of prototypical affine-invariant features, as determined by an exhaustive nearest neighbor search with the Earth Mover's Distance (EMD) developed by Rubner et al. in [92]. Invariant descriptors are clustered with Expectation-Maximization (EM) and class labels are assigned to descriptors in novel texture images, then refined with a relaxation step that uses neighborhood co-occurrence statistics from the training set. In more recent work, Lazebnik and colleagues vector-quantize features and allow features of the same discrete appearance type (or "channel") to be matched with the pyramid match algorithm described in this thesis (initially presented in [44]) operating over spatial features [68]. Their technique is a special case of the method in [44] and Chapter 3; however the authors have demonstrated novel and significant results on different features than those used in [44].

Mapping detected features to a set of global prototypes provides a way to translate unordered, variable-sized sets into single vectors, yielding a representation that is suitable for many existing vector-based learning methods and distances. The histogram of feature types is a compact and efficient way to describe the entire empirical distribution. However, such approaches assume that the space of features that will be encountered in novel images is known *a priori* when generating the prototypes, and they face the standard difficulty of properly choosing the number of cluster centers to use. Though a one-time cost, generating a reliable feature space quantization can be expensive and relies on the availability of representative data.

Moreover, to capture perceptual dissimilarity between empirical distributions, it has been shown that *bin-by-bin* measures that only compare features of the same type (e.g., L_p distance, normalized scalar product) are less robust than *cross-bin* measures that allow features from different bins to be matched to one another [92] (see Figure 2-1). Methods that cluster features on a per-example basis also must choose a quantization level and risk losing discrimination power when that level is too coarse [92, 67]. Unlike the above methods that apply VQ to obtain frequency vectors [102, 26, 68, 70], my approach represents an image with its actual set of feature vectors and makes it possible to find correspondences between features that are similar but may not share a quantization bin; that is, it allows a form of

cross-bin matching.

The feature representation I use in this work is based on a multi-resolution histogram, or *pyramid*, which is computed by binning data points into discrete regions of increasingly larger size. Single-level histograms have been used in various visual recognition systems, one of the first being that of Swain and Ballard [104], where the intersection of global color histograms was used to compare images. Pyramids have been shown to be a useful representation in a wide variety of image processing tasks, from image coding as shown by Burt and Adelson [15], to optical flow as shown by Anandan [4], to texture modeling as described by Malik and Perona [75]. However, previous image pyramids were constructed as ordered, global representations in image space; in this work I will develop a pyramid in the *feature* space to describe and compare unordered sets of vectors. See [50] for a summary of uses for multi-resolution histograms in visual recognition.

2.2.3 Computing Correspondences

Previous approaches for matching images based on their local appearance features have not utilized global correspondence solutions to measure similarity between the feature sets. However, a number of shape matching techniques require correspondences between discrete sets of features at some stage. In [40], Gold and Rangarajan develop an iterative graduated assignment graph matching method for comparing image boundary features based on a sub-optimal matching; it operates in time polynomial in the size of the feature sets. In a related approach, Chui and Rangarajan combine "softassign" and deterministic annealing to solve for correspondences between two equal-sized point sets [20]. The method requires that correspondences conform to a mapping of a given parametric family (i.e., thin plate splines).

Belongie and colleagues obtain least cost correspondences with an augmenting path algorithm in order to estimate an aligning transform between two shapes [7]. They achieve strong shape matching results with their method, though they note that the run-time does not scale well with the representation size due to the cubic complexity of solving for correspondences. Athitsos et al. use AdaBoost to learn an embedding that maps the Chamfer distance into Euclidean space, and apply it to edge images of hands to retrieve 3-D hand poses from large databases [5]. Carlsson characterizes local shape topologies with points and tangent lines and uses geometric hashing to compute the correspondence between two shapes' "order structures"; the method's combinatorial complexity practically limits its use to sets with few features [16].

Berg et al. optimize a linear bounding problem and then use local gradient descent to find explicit (approximately least-cost) correspondences between pairs of local shape features; the method requires $O(m^2 n \log n)$ time to compare two sets with n and m features [9]. For additional information about various distance metrics for shape matching and their computational complexities, please refer to [107].

The Earth Mover's Distance (EMD) is a correspondence-based distance developed by Rubner and colleagues that uses linear programming to find the optimal matching between two weighted point sets [92]. It has been shown for color- or texture-based similarity [92, 49],

and extended to allow unpenalized distribution transformations in work by Cohen and Guibas [22]. In [64] the EMD is applied to a database of silhouettes whose shock graphs are embedded into a normed space. In [37], a pseudo-metric derived from the EMD that respects the triangle inequality and positivity property is given and applied to measure shape similarity on edges.

When the inputs to the EMD are uniformly weighted sets with unequal cardinalities (or equivalently, have arbitrary weights but unequal total mass), the solution will produce a partial matching, but time exponential in the number of features is required in the worst case [92]. When the inputs have equal cardinalities (or equivalently, equal total mass), the complexity is still cubic in the number of features [71].

In this work I have developed a partial matching technique based on approximate correspondences that allows very efficient (linear-time) comparisons between sets of vectors. Unlike the voting approaches, my techniques incorporate co-occurrence feature statistics when computing a match, which is crucial for identifying similarities at the level of abstraction found in object categories and textures, as I show in Chapter 7. In contrast to approaches using vector-quantized bags of prototypical features, my approach does not require constructing a flat, discrete feature space, and thus allows cross-bin matchings between the original image data. Additionally, my work shows how we can compute matchings that accommodate the clutter and background features that are expected in images, which is not tolerated by the bag-of-prototype methods described in the previous section.

The primary distinction between my approach and previous correspondence-based approaches is the significantly lower computational demand, which translates into an ability to handle richer image descriptions. Finally, unlike any of the above techniques, I show how to perform sub-linear time image retrievals with very large databases based on local feature matchings, and have developed a matching appropriate to use as a kernel to learn categories or parameters of interest.

2.3 Learning from Sets of Features

To perform learning tasks like categorization or recognition with the set-of-features representation is also challenging and has inspired a significant amount of previous work. Many approaches in the vision literature perform recognition with local feature representations using nearest-neighbor (e.g., [7, 41, 102, 9]) or voting-based classifiers followed by an alignment step (e.g., [73, 77]). Unfortunately either choice may be impractical for large training sets, since their classification times increase steadily with the number of training examples. Shakhnarovich et al. have explored how approximate similarity search and feature selection can significantly reduce the cost of nearest-neighbor retrievals, though only for global (vector) feature representations [97].

Method	Complexity	Captures co-occurrences	Positive-definite	Model-free	Handles unequal cardinalities
Match [108]	$O(dm^2)$			x	x
Exponent match [74]	$O(dm^2)$		x	x	x
Greedy match [12]	$O(dm^2)$	x		x	x
Principal angles [115]	$O(dm^3)$	x	x		
Intermediate [13]	$O(dpm)$		x	x	x
Bhattacharyya [65]	$O(dm^3)$	x	x		x
KL-divergence [81]	$O(dm^2)$	x			x
Pyramid match	$O(dm \ \log D)$	x	x	x	x

Figure 2-2: A comparison of the properties of kernel approaches for comparing unordered sets. "Pyramid match" refers to the matching kernel developed in this work. Each method's computational cost is for computing a single kernel value. d is the feature vector dimension, m is the (maximum) set cardinality, p is the number of prototype features used in [13], and D is the aspect ratio of the vector space.

2.3.1 Kernel-Based Learning

Kernel methods, which include the Support Vector Machine (SVM), kernel Principal Components Analysis (PCA), and Gaussian Processes, have become well-established tools that are useful in a variety of contexts, including discriminative classification, regression, density estimation, and clustering (see [99, 106]). Kernel functions, which measure similarity between inputs, introduce non-linearities to the learned functions; the kernel non-linearly maps two examples from the input space to the inner product in some feature space. For example, the SVM is a widely used approach to discriminative classification that finds the optimal separating hyperplane between two classes. Whereas nearest-neighbor classifiers require pairwise comparisons between a novel example and all training examples, an SVM identifies a sparse subset of the training examples (the support vectors), and the novel example must be compared against only these examples.

However, conventional kernels (such as the Gaussian RBF or polynomial) are designed to operate on $\Re^n$ vector inputs, where each vector entry corresponds to a particular global attribute for that instance. As a result, initial approaches using SVMs for recognition were forced to rely on global image features [18, 91, 86], which have limitations under real-world conditions as discussed above.

More recently, attention has been focused on developing specialized kernels that can more fully leverage kernel-based learning algorithms for situations where the data are most naturally represented by something other than a Euclidean vector space, such as graphs, strings, or trees [36]. Due to the increasing prevalence of data that is best represented by local features, several researchers have designed kernels that operate on unordered sets of vectors [74, 65, 115, 98, 12, 108, 27, 81, 13, 66]. Existing techniques generally require either solving for explicit correspondences between features or fitting a known parametric distribution to each set in order to evaluate an existing distribution-based similarity metric. Each kernel has its own strengths and weaknesses, and only a few have been applied in computer vision; see Figure 2-2 for a concise comparison, and below for elaboration.

Current approaches are either prohibitively computationally expensive, make strong assumptions regarding the parametric form of the features, discard information by replacing inputs with prototypical features, ignore co-occurrence information by considering features independently, are not positive-definite, and (or) are limited to sets of equal size. My work overcomes a number of these limitations. The matching measure I develop in this thesis forms a Mercer (positive semi-definite) kernel suitable for use with the input space of sets of vectors within any existing kernel-based learning paradigm (see Section 3.4 in the next chapter).

Parametric Kernels

Kondor and Jebara build a parametric kernel, where a Gaussian is fit to each set of vectors, and then the kernel value between two sets is the Bhattacharyya affinity between their Gaussian distributions [65] . The authors note that the method is constrained to using a Gaussian model in order to have a closed form solution. The Bhattacharyya kernel shows strong performance in a study by Eichhorn and Chapelle [30], though it is also noted that in practice the kernel is limited to sets with small cardinalities, given its cubic complexity. Moreno et al. fit a Gaussian to a feature set, and then compare sets using KL-divergence as a distance measure [81].

Approaches which fit a parametric model to feature sets in order to compare their distributions [65, 81, 27] can be computationally costly and may have limited applicability, since they assume both that features within a set will conform to the chosen distribution, and that sets will be adequately large enough to extract an accurate estimate of the distribution's parameters. These assumptions are violated regularly by real data, which will often exhibit complex variations within a single bag of features (e.g., patches from an image), and will produce a wide range of cardinalities per instance (e.g., titles of documents have just a few word features). I instead take a non-parametric, "model-free" approach, representing bags of features directly with multi-dimensional, multi-resolution histograms.

Matching Kernels

Kernel methods that use explicit correspondences between two sets' features search one set for the best matching feature for each member in the other, and then define set similarity as a function over those component similarity values [108, 74, 12, 13]. Wallraven et al. propose a kernel that averages over the similarities of the best matching feature found for each feature member within the other set, though the kernel is non-Mercer and so lacks convergence guarantees when used in an SVM [108]. A similar kernel given by Lyu also considers all possible feature matchings but raises the similarity between each pair of features to a given power [74]. The method given by Boughorbel and colleagues is based on finding a sub-optimal matching between two sets using a greedy heuristic; although this results in a non-Mercer kernel, the authors provide a means of tuning the kernel hyperparameter so as to limit the probability that a given kernel matrix is not positive-definite [12]. In more recent work by Boughorbel et al., correspondences are found through the use of an intermediate set of prototype features in order to insure positive-definiteness [13].

These matching kernel methods have complexities that are quadratic in the number of features m, hindering usage for kernel-based learning when feature sets are large. The "intermediate" matching kernel of Boughorbel and colleagues [13] has a quadratic run-time if the number of prototypes $p = O(m)$. That is reduced if p is set so that $p < m$; however, the authors note that higher values of p yield more accurate results. Furthermore, matching each input feature independently as in [108, 74, 13] assumes non-injective correspondences and, similar to the voting approaches, ignores potentially useful information about intra-set dependencies. In contrast, the set kernel I develop in this work captures the joint statistics of co-occurring features by matching them concurrently as a set.

Other Kernels for Local Features

In the method of Wolf and Shashua, similarity is measured in terms of the principal angle between the linear subspaces spanned by two sets' vector elements; the kernel has a cubic complexity and is only positive-definite for sets of equal cardinality [115]. In work by Shashua and Hazan, an algebraic kernel is used to combine similarities given by local (vector-based) kernels, with the weighting chosen to reflect whether the features are in alignment (ordered) [98]. When set cardinalities vary, inputs are padded with zeros so as to form equal-size matrices; results are only shown for a classification task with input sets whose features' ordering is known. Holub and colleagues use a hybrid generative-discriminative approach for object recognition [53], combining the Fisher kernel of Jaakkola and Haussler [58] and a probabilistic "constellation model" (see below).

2.3.2 Alternative Learning and Detection Methods

An alternative approach to kernel methods for recognition when dealing with unordered set data is to designate prototypical examples from each class, and then represent examples by a vector giving their distances to each prototype; standard algorithms that handle vectors in a Euclidean space are then applicable. The issues faced by such a prototype-based method are determining which examples should serve as prototypes, choosing how many there should be, and updating the prototypes properly when new types of data are encountered. Zhang and Malik build such a classifier for handwritten digits, and use Belongie et al.'s shape context distance as the measure of similarity [120]. To reduce the complexity of training an SVM from a large number of examples, in recent work Zhang et al. suggest a combination of nearest-neighbor search and SVMs [119].

Agarwal and Roth have developed a sparse part-based approach to object detection for objects having a relatively fixed spatial configuration of parts [3]. Quattoni and colleagues develop a discriminative part-based approach to object recognition based on Conditional Random Fields with hidden variables, and show how to relax a conditional independence assumption of local features [88].

In [32], Felzenszwalb and Huttenlocher develop an efficient method to detect and localize an object composed of some number of defined parts in a given deformable configuration within an image. Using dynamic programming and generalized distance transforms, they compute the globally optimal matching of the tree-structured part-based model to a set of locations

in the image. The method relies on the assumptions that the relationships between the parts form a tree structure, and that relationships between parts are pairwise. Under the first assumption, the result is an $O(l^2m)$ time matching to image locations, where there are m parts and l possible locations for each part. Under both assumptions, only $O(lm)$ time is required. In practice, generally l is quite large and $l >> m$.

In a related approach, Crandall and colleagues develop a family of spatial priors called k-fans that allow varying degrees of spatial structure between parts to be captured [24]. An object model is represented by some number (k) of reference parts, and the position of all other parts in the model are evaluated relative to these reference parts. To localize a model in an image with l possible locations for each part, the running time is $O(lm^{k+1})$. For a Gaussian k-fan model, $O(lm^k)$ time is required. The authors of [24] find that for several object classes used in the literature, relatively small amounts of spatial structure accomplish a detection accuracy comparable to that achieved with more complex spatial models.

Note that unlike techniques designed to measure matching quality between sets of unstructured local features, the matchings computed in [32] and [24] consider both the parts' appearance as well as the pairwise relationships between their positions; additionally, rather than evaluating similarity for recognition purposes, these techniques are designed for the detection task of localizing a given configuration of parts within an image.

A number of generative learning techniques have also been proposed for object recognition. The "constellation model" developed by Burl et al. models a class of objects in terms of a fixed number of parts and those parts' spatial relationships [14]. A means of learning the model with weak supervision was presented by Weber and colleagues [111, 110]. Fergus et al. extended the approach to account for appearance and scale variability [34], while Fei-Fei et al. explored Bayesian priors over the model parameters to make learning from small amounts of examples more practical [31]. The exponential complexity of these models has limited their use to very small numbers of parts (on the order of 5).

2.4 Approximate Matching Algorithms

The optimal matching's cubic complexity, though polynomial, is still costly enough that it can be prohibitively expensive in practice, and often forces researchers relying on computing a matching to artificially limit the size of the sets they allow their method to consider. However, researchers in the theory community have developed approximation algorithms to compute solutions close to the optimal bijective matching for a fraction of the computational cost [57, 19, 2] (and see the survey on earlier related techniques in [6]).

My work draws inspiration from these methods, particularly regarding their use of a hierarchical decomposition of the feature space. However, previous matching approximations are restricted to equal-sized set inputs, meaning that partial matchings are not possible. This constraint is typically not satisfied in the vision domain, where a widely varying number of local parts may be detected or extracted from each image. When all points in both sets are forced into correspondence, the measure can be intolerant to clutter and outlier features. In contrast, my techniques enable partial matchings and allow us to use inputs having unequal

cardinalities, which is important in practice for handling clutter and unsegmented images.

In the method of Indyk and Thaper, multi-resolution histograms are compared with the L_1 distance to approximate a least-cost matching of equal-sized sets, and the measure is applied to perform approximate nearest-neighbor image retrievals based on global color histograms [57]. This work inspired me to use a similar representation for point sets and to consider counting matches within histograms. However, unlike the method in [57], the approximate matching I have designed compares histograms with a weighted intersection rather than L_1, and more significantly, it forms a kernel that can build kernel-based classifiers and regressors over sets of local features, and it computes partial matchings between inputs that may have variable sizes.

In addition, previous matching approximations suffer from distortion factors that scale linearly with the feature dimension, and they fail to take advantage of structure in the feature space. In Chapter 4 I show how to alleviate this decline in accuracy for high-dimensional features by tuning the hierarchical decomposition according to the particular structure of the data, when such structure exists.

Chapter 3

The Pyramid Match

In this chapter I introduce the partial matching approximation called the *pyramid match*. I begin in Section 3.1 by laying out some definitions regarding the input space of point sets and the meaning of an optimal matching. After these preliminaries, I define the pyramid match algorithm in Section 3.2. I provide an analysis of its complexity in Section 3.3, proof that it forms a Mercer kernel in Section 3.4, and proof of the expected error bounds in Section 3.5. Applications and results using this measure within the context of vision problems are presented in later chapters.

3.1 Preliminaries

I will be considering a feature space F of d-dimensional vectors. The point sets (or multi-sets, since duplications of features may occur within a single set) we match will come from the input space S, which contains sets of feature vectors drawn from F:

$$S = \Big\{ \mathbf{X} | \mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \Big\}, \tag{3.1}$$

where each feature is a d-dimensional vector, $\mathbf{x}_i \in F \subseteq \Re^d$, and $m = |\mathbf{X}|$. Note that the point dimension d is fixed for all features in F, but the value of m may vary across instances in S. The values of elements in vectors in F have a maximal range (aspect ratio) D, and the minimum inter-vector distance between unique points is 1, which may be enforced by scaling the data to some precision and truncating to integer values. Please note that throughout the text I will use the terms feature, vector, and point interchangeably to refer to the elements within a set.

In this work we want to efficiently approximate the optimal partial matching. A partial matching between two point sets is an assignment that maps all points in the smaller set to some subset of the points in the larger (or equal-sized) set. Given point sets $\mathbf{X}$ and $\mathbf{Y}$, where $m = |\mathbf{X}|$, $n = |\mathbf{Y}|$, and $m \leq n$, a partial matching

$$\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi) = \{(\mathbf{x}_1, \mathbf{y}_{\pi_1}), \ldots, (\mathbf{x}_m, \mathbf{y}_{\pi_m})\} \tag{3.2}$$

pairs each point in $\mathbf{X}$ to some unique point in $\mathbf{Y}$ according to the permutation of indices specified by $\pi = [\pi_1, \ldots, \pi_m]$, $1 \leq \pi_i \leq n$, where π_i specifies which point $\mathbf{y}_{\pi_i} \in \mathbf{Y}$ is matched to $\mathbf{x}_i \in \mathbf{X}$, for $1 \leq i \leq m$. The cost of a partial matching is the sum of the distances between matched points:

$$\mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)\right) = \sum_{\mathbf{x}_i \in \mathbf{X}} ||\mathbf{x}_i - \mathbf{y}_{\pi_i}||_1. \tag{3.3}$$

The optimal partial matching $\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)$ uses the assignment π^* that minimizes this cost:

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\, \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)\right). \tag{3.4}$$

In this work I also consider the partial matching similarity score, with similarity measured in terms of inverse distance or cost. The similarity $\mathcal{S}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)\right)$ of a partial matching is the sum of the inverse distances between matched points:

$$\mathcal{S}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)\right) = \sum_{\mathbf{x}_i \in \mathbf{X}} \frac{1}{||\mathbf{x}_i - \mathbf{y}_{\pi_i}||_1 + 1}, \tag{3.5}$$

where the distance in the denominator is incremented by 1 to avoid division by zero. We define the optimal partial matching similarity score to be the similarity resulting from the same matching that minimizes the cost (Eqn. 3.4).

The best matching between two sets corresponds to an assignment that produces the minimal summed cost (distance) between component features. This matching assumes that two vectors' proximity in the chosen feature space $F \subseteq \Re^d$ reflects their similarity, and that the overall set (dis)similarity is well-captured by this aggregation of feature-level scores. However, note that in contrast to a matching that sums the similarity between the nearest feature for every feature in a set, the optimal matching is one-to-one.

3.2 Pyramid Match Algorithm

The pyramid match approximation uses a multi-dimensional, multi-resolution histogram pyramid to partition the feature space into increasingly larger regions. At the finest resolution level in the pyramid, the partitions (bins) are very small; at successive levels they continue to grow in size until the point where a single partition encompasses the entire feature space. At some level along this gradation in bin sizes, any two points from any two point sets will begin to share a bin in the pyramid, and when they do, they are considered matched. The pyramid allows us to extract a matching score without computing distances between any of the points in the input sets—when points sharing a bin are counted as matched, the size of that bin indicates the farthest distance any two points in it could be from one another.

Each feature set is mapped to a multi-resolution histogram that preserves the individual features' distinctness at the finest level. The histogram pyramids are then compared using a weighted histogram intersection computation, which I show defines an implicit correspondence based on the finest resolution histogram cell where a matched pair first appears (see

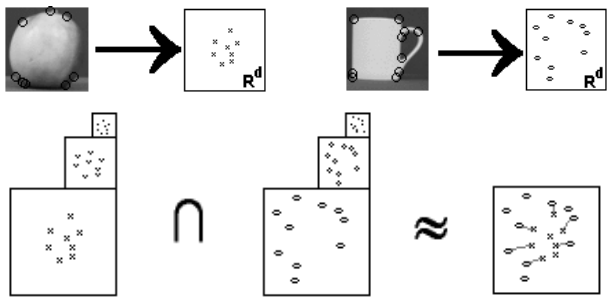


Figure 3-1: The pyramid match intersects histogram pyramids formed over sets of features, approximating the optimal correspondences between the sets' features. Note throughout that the points in the feature space will typically *not* be 2-D image coordinates; rather, they may be any d-dimensional descriptions of local shape or appearance.

Figure 3-1). The computation time of both the pyramids themselves as well as the weighted intersection is linear in the number of features.[1]

The feature extraction function Ψ for an input set $\mathbf{X}$ is defined as:

$$\Psi(\mathbf{X}) = [H_0(\mathbf{X}), \ldots, H_{L-1}(\mathbf{X})], \tag{3.6}$$

where $\mathbf{X} \in S$, $L = \lceil \log_2 D \rceil + 1$, $H_i(\mathbf{X})$ is a histogram vector formed over points in $\mathbf{X}$ using d-dimensional bins of side length 2^i, and $H_i(\mathbf{X})$ has a dimension $r_i = \left(\frac{D}{2^i}\right)^d$. In other words, $\Psi(\mathbf{X})$ is a histogram pyramid, where each subsequent component histogram has bins that double in size (in all d dimensions) compared to the previous one. The bins in the finest-level histogram H_0 are small enough that each unique d-dimensional data point from features in F falls into its own bin, and then the bin size increases until all points in F fall into a single bin at level $L-1$.

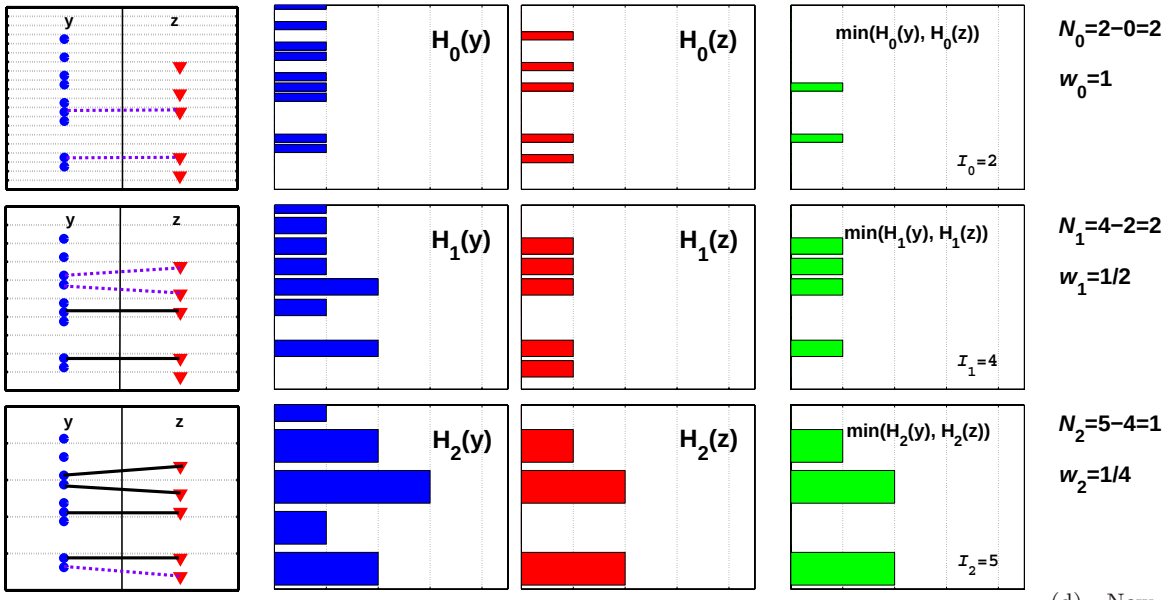
The pyramid match $\mathcal{P}_\Delta$ measures similarity (or dissimilarity) between point sets based on implicit correspondences found within this multi-resolution histogram space. The similarity between two input sets $\mathbf{Y}$ and $\mathbf{Z}$ is defined as the weighted sum of the number of feature matchings found at each level of the pyramid formed by Ψ:

$$\mathcal{P}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = \sum_{i=0}^{L-1} w_i N_i, \tag{3.7}$$

where N_i signifies the number of newly matched pairs at level i, and w_i is a weight for matches formed at level i (and will be defined below). A new match is defined as a pair of features that were not in correspondence at any finer resolution level.

The matching approximation implicitly finds correspondences between point sets, if we consider two points matched once they fall into the same histogram bin, starting at the finest resolution level where each unique point is guaranteed to be in its own bin. The correspondences are *implicit* in that matches are counted and weighted according to their strength, but the specific pairwise links between points need not be individually enumerated. The matching is a hierarchical process: vectors not found to correspond at a high resolution

[1] I first reported the basic pyramid match algorithm in [44].



(a) Point sets (b) Histogram pyramids (c) Intersections (d) New matches

Figure 3-2: The pyramid match ($\mathcal{P}_\Delta$) determines a partial correspondence by matching points once they fall into the same histogram bin. In this example, two 1-D feature sets are used to form two histogram pyramids. Each row corresponds to a pyramid level. In (a), the set $\mathbf{Y}$ is on the left side, and the set $\mathbf{Z}$ is on the right. (Points are distributed along the vertical axis, and these same points are repeated at each level.) Light dotted lines are bin boundaries, bold dashed lines indicate a new pair matched at this level, and bold solid lines indicate a match already formed at a finer resolution level. In (b) multi-resolution histograms are shown, with bin counts along the horizontal axis. In (c) the intersection pyramids between the histograms in (b) are shown. $\mathcal{P}_\Delta$ uses these intersection counts to measure how many new matches occurred at each level. Here, $\mathcal{I}_i = \mathcal{I}(H_i(\mathbf{Y}), H_i(\mathbf{Z})) = 2, 4, 5$ across levels, and therefore the number of new matches found at each level are $N_i = 2, 2, 1$. ($\mathcal{I}_{-1} = 0$ by definition.) The sum over N_i, weighted by $w_i = 1, \frac{1}{2}, \frac{1}{4}$, gives the pyramid match similarity.

have the opportunity to be matched at lower resolutions. For example, in Figure 3-2, there are two points matched at the finest scale, two new matches at the medium scale, and one at the coarsest scale. $\mathcal{P}_\Delta$'s output value reflects the overall similarity of the matching: each newly matched pair at level i contributes a value w_i that is proportional to how similar two points matching at that level must be, as determined by the bin size.

To calculate N_i, the pyramid match makes use of a histogram intersection function $\mathcal{I}$, which measures the "overlap" between two histograms' bin counts:

$$\mathcal{I}\left(\mathbf{A}, \mathbf{B}\right) = \sum_{j=1}^{r} \min\left(\mathbf{A}^{(j)}, \mathbf{B}^{(j)}\right), \tag{3.8}$$

where $\mathbf{A}$ and $\mathbf{B}$ are histograms with r bins, and $\mathbf{A}^{(j)}$ denotes the count of the j^{th} bin of $\mathbf{A}$.

Histogram intersection effectively counts the number of points in two sets that match at a given quantization level, i.e., fall into the same bin. To calculate the number of newly matched pairs N_i induced at level i, it is sufficient to compute the difference between successive histogram levels' intersections:

$$N_i = \mathcal{I}\left(H_i(\mathbf{Y}), H_i(\mathbf{Z})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{Y}), H_{i-1}(\mathbf{Z})\right), \tag{3.9}$$

where H_i refers to the i^{th} component histogram generated by Ψ in Eqn. 3.6. Note that the measure is not searching explicitly for similar points – it never computes distances between the vectors in each set. Instead, it simply uses the change in intersection values at each histogram level to count the matches as they occur. In addition, due to the subtraction in Eqn. 3.9, the output score will reflect an underlying matching that is one-to-one.

The sum in Eqn. 3.7 starts with index $i = 0$ because by the definition of F, the same points that could match at level 0 can match at (a hypothetical) level -1. Therefore we can start collecting new match counts at level 0, and define level -1 to be a base case with no intersections: $\mathcal{I}\left(H_{-1}(\mathbf{Y}), H_{-1}(\mathbf{Z})\right) = 0$. All matches formed at level 0 are new.

The number of new matches found at each level in the pyramid is weighted according to the size of that histogram's bins: to measure similarity, matches made within larger bins are weighted less than those found in smaller bins. Specifically, a geometric estimate of the distance between any two points sharing a particular bin is made; in terms of L_1 cost, two points in the same bin can only be as far from one another as the sum of the lengths of the bin's sides. At level i in a pyramid, this length is equal to $d2^i$. Thus, the number of new matches induced at level i is weighted by $w_i = \frac{1}{d2^i}$ to reflect the (worst-case) similarity of points matched at that level. Intuitively, this means that similarity between vectors (features in $\mathbf{Y}$ and $\mathbf{Z}$) at a finer resolution—where features are more distinct—is rewarded more heavily than similarity between vectors at a coarser level.[2] Moreover, as I show in Section 3.5, this particular setting of the weights enables us to prove theoretical error bounds for the pyramid match cost.

From Eqns. 3.7 and 3.9, I define the (un-normalized) pyramid match:

$$\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = \sum_{i=0}^{L-1} w_i \Big(\mathcal{I}\left(H_i(\mathbf{Y}), H_i(\mathbf{Z})\right) - \mathcal{I}(H_{i-1}(\mathbf{Y}), H_{i-1}(\mathbf{Z})) \Big), \tag{3.10}$$

where $\mathbf{Y}, \mathbf{Z} \in S$, and $H_i(\mathbf{Y})$ and $H_i(\mathbf{Z})$ refer to the i^{th} histogram in $\Psi(\mathbf{Y})$ and $\Psi(\mathbf{Z})$, respectively.

3.2.1 Matching Score Normalization

To provide a partial matching score that does not favor large input sets and has no penalty whatsoever for the unmatched features, we can normalize this value by the cardinality of

[2]This is if the pyramid match is measuring similarity; an inverse weighting scheme is applied if the pyramid match is measuring cost. To use the matching as a cost function, weights are set as the distance estimates ($w_i = d2^i$); to use as a similarity measure, weights are set as (some function of) the inverse of the distance estimates ($w_i \propto \frac{1}{d2^i}$).

the smaller of the two input sets, arriving at the final output value

$$\mathcal{P}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = \frac{\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right)}{\min\left(|\mathbf{Y}|, |\mathbf{Z}|\right)}. \tag{3.11}$$

Alternatively, when measuring similarity, both sets' sizes may be taken into account by normalizing by the product of each input's self-similarity, for a final kernel value

$$\mathcal{P}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = \frac{\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right)}{\sqrt{\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Y})\right) \times \tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Z}), \Psi(\mathbf{Z})\right)}}. \tag{3.12}$$

The normalization choice depends on whether or not the presence of greater numbers of unmatchable features should diminish the matching quality, and whether or not a metric measure is desired. In Chapter 5 I will show that the product normalization is necessary to ensure a metric that can be computed within a randomized hashing framework.

3.2.2 Pyramid Bin Structure

In order to alleviate quantization effects that may arise due to the discrete histogram bins, we combine the values resulting from multiple (T) pyramid matches formed under different multi-resolution histograms with randomly shifted bins. Each dimension of each of the T pyramids is shifted by an amount chosen uniformly at random between 0 and D. This yields T feature mappings $\Psi_1, \ldots, \Psi_T$ that are applied as in Eqn. 3.6 to map an input set $\mathbf{Y}$ to T multi-resolution histograms: $[\Psi_1(\mathbf{Y}), \ldots, \Psi_T(\mathbf{Y})]$. If measuring cost, the matching value for inputs $\mathbf{Y}$ and $\mathbf{Z}$ is then $\min_{j=1,\ldots,T} \mathcal{P}_\Delta\left(\Psi_j(\mathbf{Y}), \Psi_j(\mathbf{Z})\right)$, the least-cost matching found across all translations for the two sets.

In fact, the rationale for considering multiple random shifts of the pyramid grids is not only to satisfy the intuitive need to reduce quantization effects caused by bin placements; it also allows us to theoretically measure how probable it is that any two points will be separated by a bin boundary, as I describe below in Section 3.5. When dealing with single-dimensional histograms, researchers will often employ overlapping histogram bins, or use soft bin assignment (for instance, see [1, 103, 72]). This could have similar benefits for the *accuracy* of the pyramid match, but it is simply not computationally practical for higher-dimensional pyramids. It would cost time exponential in the dimension d of the feature space in order to cover the combinations of overlap or soft assignment between the d-dimensional bins when inserting a single point. In contrast, the increase in cost to apply a pyramid shift remains linear in d.

To insert a point into a histogram having uniformly shaped bins, the bin index is computed by dividing each coordinate of the point by the bin size. To insert a point into a histogram whose boundaries are translated by a given amount, the shift amount in each dimension is subtracted from the feature value of the point in the corresponding dimension before that division.

To further refine a pyramid match, multiple pyramids with unique initial (finest level) bin sizes may be used. The set of unique bin sizes produced throughout a pyramid determines

the range of distances at which features will be considered for possible matchings. With bin sizes doubling in size at every level of a pyramid, this means that the range of distances considered will rapidly increase until the full diameter of the feature space is covered. However, by incorporating multiple pyramids with unique initial bin sizes, we can boost that range of distances to include more diverse gradations. This is accomplished by setting the side length for each histogram $H_i(\mathbf{X})$ to $f2^i$, where f is the side length at the finest resolution (in the above definition, $f = 1$). For example, say $D = 32$, so $L = 6$. Then the set of distances considered with $f = 1$ are $\{1, 2, 4, 8, 16, 32\}$; adding a second pyramid with $f = 3$ increases this set to also consider the distances $\{3, 6, 12, 24\}$. The outputs from pyramids with multiple starting side lengths can be combined in the same way that the outputs from pyramids with multiple translations are combined.

Notice that the bins in the pyramids (and thus the distances at which matches are considered) would also grow more gradually if we were to select an alternative factor of bin size increase between 1 and 2. However, a non-integer factor would not preserve the parent-child relationship between bins, and thus the pyramid match counting of new matches would not be possible. In other words, points that matched at level i would no longer be guaranteed to match again at level $i + 1$, and so the number of new matches N_i could not be efficiently counted by the difference in histogram intersections as I have outlined above. In Chapter 4 I present a method for tuning the pyramid bin boundaries according to the underlying structure of the feature space, which allows the bin sizes to grow more gradually across pyramid levels while still maintaining the parent-child relationship.

3.2.3 Explicit Approximate Correspondence Fields

In addition to scalar matching scores, we can optionally use the pyramid structure to extract explicit correspondence fields by considering the possible matchings between points that cause a nonzero intersection value in a given bin. In this case, the pyramid decomposes the required matching computation into a hierarchy of smaller matchings. Upon encountering a bin with a nonzero intersection, an optimal matching is computed between only those features from the two sets that fall into that particular bin. All points that are used in that per-bin matching are then flagged as matched and may not take part in subsequent matchings at coarser resolutions of the pyramid.

In Chapter 6 I provide experiments with this technique to produce approximate correspondence fields between image features; I also show that with pyramids tuned to the feature space distribution, the matching computed within a bin need not be optimal to produce similar aggregate results. In Chapter 9 I develop a method for automatically recovering category feature masks from unlabeled images that uses the approximate correspondences to identify consistently matched image features.

3.2.4 Partial Match Correspondences

The pyramid match allows input sets to have unequal cardinalities, and therefore it enables partial matchings, where the points of the smaller set are mapped to some subset of the points in the larger set. Dissimilarity is only judged on the most similar part of the empirical

distributions, and superfluous data points from the larger set are entirely ignored; the result is a robust similarity measure that accommodates inputs expected to contain extraneous vector entries. This is a common situation when recognizing objects in images, due for instance to background variations, clutter, or changes in object pose that cause different subsets of features to be visible. As a kernel, the proposed matching measure is equipped to handle unsegmented or poorly segmented examples, as I will demonstrate in Chapter 8.

Since the pyramid match defines correspondences across entire sets simultaneously, it inherently accounts for the distribution of features occurring in one set. In contrast, previous approaches have used each feature in a set to independently index into the second set; this ignores possibly useful information that is inherent in the co-occurrence of a set of distinctive features, and it fails to distinguish between instances where an object has varying numbers of similar features since multiple features may be matched to a single feature in the other set [108, 74, 13, 73, 77, 105, 95].

Note that the optimal partial matching defined in Section 3.1 is most appropriate when the smaller set is a group of model features, the larger set is a group of observed features, and we wish to match the model's features onto some part of the observed features. However, it is also possible with the pyramid match to measure the matching score using only a subset of *both* input sets' features. For such a matching to be well-defined, it is necessary to provide a limit t on how distant points included in a matching can be. Given this limit, a matching can be "partial" in both directions, meaning that only a subset of either sets' features might contribute to the score. The corresponding modified matching cost function $\mathcal{C}'$ is defined as:

$$\mathcal{C}'\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)\right) = \sum_{\pi_i \in \pi, s.t. ||\mathbf{x}_i - \mathbf{y}_{\pi_i}|| < t} ||\mathbf{x}_i - \mathbf{y}_{\pi_i}||. \tag{3.13}$$

To implement this cost function with the pyramid match, fewer than L levels in the pyramids are considered in the sum in Eqn. 3.10. Specifically, new matches are only counted for the levels where the bin sizes are smaller than t.

3.3 Complexity Analysis

A key aspect of this method is that we obtain a measure of matching quality between two point sets without computing pairwise distances between their features—an $O(m^2)$ savings over sub-optimal greedy matchings. Instead, we exploit the fact that the points' placement in the pyramid reflects their distance from one another in the feature space.[3]

The time required to compute the L-level histogram pyramid $\Psi(\mathbf{X})$ for an input set with $m = |\mathbf{X}|$ d-dimensional features is $O(dzL)$, where $z = \max(m, k)$ and k is the maximum histogram index value in a single dimension. For a histogram at level i, $k \leq \frac{D}{2^i}$, so at any level, $k \leq D$. (Typically $m > k$.) The bin coordinates corresponding to nonzero histogram entries for each of the $L = \lceil \log_2 D \rceil + 1$ quantization levels are computed directly during a scan of the m input vectors; these entries are sorted by the bin indices and the bin counts for all entries with the same index are summed to form one entry. This sorting requires only

[3]Note throughout the text that the distance between points or features refers to distance in the d-dimensional feature space F, and generally F will *not* be the image space or 2-D spatial coordinates.

$O(dm + dk)$ time using the radix-sort algorithm with counting sort, a linear time sorting algorithm that is applicable to the integer bin indices [23]. The histogram pyramid that results is high-dimensional, but very sparse, with only $O(mL)$ nonzero entries that need to be stored.

The computational complexity of $\mathcal{P}_\Delta$ is $O(dmL)$, since computing the intersection values for histograms that have been sorted by bin index requires time linear in the number of nonzero entries (*not* the number of actual bins). With sorted bin index lists, we obtain the intersection value by running one pointer down each of the two lists; one pointer may not advance until the other is incremented down to an index at least as high as the other, or else the end of its list. Then, whenever the two lists share a nonzero index, the intersection count is incremented by the minimum bin count between the two. If one list has a nonzero count for some bin index but the other has no entry for that index, the minimum count is 0, so no update is needed.

Generating multiple pyramid matches with randomly shifted grids scales the complexity by T, the constant number of shifts. (Typically we will use $1 \leq T \leq 3$.) All together, the complexity of computing both the pyramids and kernel or cost values is $O(TdmL)$. In contrast, the optimal matching requires $O(dm^3)$ time, which severely limits the practicality of large input sizes. Refer back to Figure 2-2 in Chapter 2 for complexity comparisons with existing set kernels.

3.4 The Pyramid Match as a Mercer Kernel

Kernel-based learning algorithms are founded on the idea of embedding data into a Euclidean space, and then seeking linear relations among the embedded data [99, 106]. For example, a Support Vector Machine (SVM) finds the optimal separating hyperplane between two classes in an embedded space (also referred to as the feature space). A kernel function $K : X \times X \rightarrow \Re$ serves to map pairs of data objects in an input space X to their inner product in the embedding space E, thereby evaluating the similarities between all data objects and determining their relative positions. Linear relations are sought in the embedded space, but a decision boundary may still be non-linear in the input space, depending on the choice of a feature mapping function $\Phi : X \rightarrow E$.

Only positive semi-definite kernels guarantee an optimal solution to kernel-based algorithms based on convex optimization, which includes SVMs. According to Mercer's theorem, a kernel K is positive semi-definite if and only if there exists a mapping Φ such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \ \forall \mathbf{x}_i, \mathbf{x}_j \in X, \tag{3.14}$$

where $\langle \cdot, \cdot \rangle$ denotes a scalar dot product. This insures that the kernel corresponds to an inner product in some feature space, where kernel methods can search for linear relations [99].

Histogram intersection on single resolution histograms over multi-dimensional data was shown to be a positive-definite similarity function by Odone et al. in [86]. That is, the intersection $\mathcal{I}(H(\mathbf{Y}), H(\mathbf{Z}))$ is a Mercer kernel. The proof shows that there is an explicit feature mapping after which the intersection is an inner product. Specifically, the mapping

$\mathcal{V}$ encodes an r-bin histogram H as a p-dimensional binary vector, $p = m \times r$, where m is the total number of points in the histogram:

$$\mathcal{V}(H) = \left(\underbrace{\overbrace{1, \ldots, 1}^{H^{(1)}}, \overbrace{0, \ldots, 0}^{m - H^{(1)}}}_{\text{first bin}}, \ldots, \underbrace{\overbrace{1, \ldots, 1}^{H^{(r)}}, \overbrace{0, \ldots, 0}^{m - H^{(r)}}}_{\text{last bin}} \right). \tag{3.15}$$

The inner product between the binary strings output from $\mathcal{V}$ is equivalent to the original histograms' intersection value [86]. If m varies across examples, the above holds by setting $p = M \times r$, where M is the maximum size of any input. Note that this binary encoding only serves to prove positive-definiteness and is never computed explicitly.

Using this proof and the closure properties of valid kernel functions, I can show that the pyramid match kernel is a Mercer kernel. Recall that for a kernel (similarity) value the weights for the pyramid match intersections are set to be inversely proportional to the bin size. The definition given in Eqn. 3.10 is algebraically equivalent to

$$\tilde{\mathcal{P}}_{\Delta}\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = w_{L-1}\, \mathcal{I}\left(H_{L-1}(\mathbf{Y}), H_{L-1}(\mathbf{Z})\right) + \sum_{i=0}^{L-2} (w_i - w_{i+1})\, \mathcal{I}\left(H_i(\mathbf{Y}), H_i(\mathbf{Z})\right), \tag{3.16}$$

since $\mathcal{I}\left(H_{-1}(\mathbf{Y}), H_{-1}(\mathbf{Z})\right) = 0$ by definition. Given that Mercer kernels are closed under both addition and scaling by a positive constant [99], the above form shows that the pyramid match kernel is positive semi-definite for any weighting scheme where $w_i \geq w_{i+1}$. In other words, if we can insure that the weights decrease for coarser pyramid levels, then the pyramid match will sum over positively weighted positive-definite kernels, yielding another positive-definite kernel. Using the weights $w_i = \frac{1}{d2^i}$, we do maintain this property. The sum combining the outputs from multiple pyramid matches under different random bin translations likewise remains Mercer. Therefore, the pyramid match is valid for use as a kernel in any existing learning methods that require Mercer kernels.

In Section 8.1.2 of Chapter 8 I will discuss how to adjust pyramid match kernel matrices whose diagonal entries are very large relative to their off-diagonal entries, which may be necessary for best performance when used with some kernel methods, including Support Vector Machines.

3.5 Approximation Error Bounds

In this section I show theoretical approximation bounds on the *cost* measured by the pyramid match relative to the cost measured by the optimal partial matching defined in Section 3.1. Recall the cost (as opposed to similarity) is measured by setting $w_i = d2^i$ in Eqn. 3.10. In earlier work, the authors of [57] provided bounds for a multi-resolution histogram embedding's approximation of the optimal matching between inputs with equal cardinality. Some of the main ideas of the proofs below are similar, but they have been adapted and extended to show the expected error bounds for partial matchings, where input sets can have variable numbers of features, and some features do not affect the matching cost.

Lemma 1

For any two point sets $\mathbf{X}$,$\mathbf{Y}$ where $|\mathbf{X}| \leq |\mathbf{Y}|$, we have

$$\mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right) \leq \mathcal{P}_\Delta\left(\mathbf{X}, \mathbf{Y}\right).$$

Proof:

Consider a matching induced by pairing points within the same cells of each histogram H_i. The cost induced by matching two points within a bin having d sides that are each of length 2^i is at most $d2^i$ under the L_1 ground distance.

There are no pairings induced by the histograms at level -1, or $\mathcal{I}\left(H_{-1}(\mathbf{X}), H_{-1}(\mathbf{Y})\right) = 0$. At level 0 there are $\mathcal{I}\left(H_0(\mathbf{X}), H_0(\mathbf{Y})\right)$ pairs of points that can be matched together within the same bins of H_0, which induces a cost no more than $d\,\mathcal{I}\left(H_0(\mathbf{X}), H_0(\mathbf{Y})\right)$. Then $\mathcal{I}\left(H_1(\mathbf{X}), H_1(\mathbf{Y})\right) - \mathcal{I}\left(H_0(\mathbf{X}), H_0(\mathbf{Y})\right)$ new pairs of points are matched at level 1, which induces a cost no more than $2d\left[\mathcal{I}\left(H_1(\mathbf{X}), H_1(\mathbf{Y})\right) - \mathcal{I}\left(H_0(\mathbf{X}), H_0(\mathbf{Y})\right)\right]$, and so on.

In general, histogram H_i induces cost $d2^i\left[\mathcal{I}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})\right)\right]$. Summing the costs induced by all levels, we have:

$$\begin{aligned} \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right) &\leq \sum_{i=0}^{L-1} d2^i \Big(\mathcal{I}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})\right)\Big) \\ &\leq \mathcal{P}_\Delta(\mathbf{X}, \mathbf{Y}) \end{aligned} \tag{3.17}$$

where $w_i = d2^i$ in Eqn. 3.10.

Lemma 2

There is a constant C such that for any two point sets $\mathbf{X}$ and $\mathbf{Y}$, where $|\mathbf{X}| \leq |\mathbf{Y}|$, if the histogram bin boundaries are shifted randomly in the same way when computing $\Psi(\mathbf{X})$ and $\Psi(\mathbf{Y})$, then the expected value of the pyramid match cost is bounded:

$$E[\mathcal{P}_\Delta(\mathbf{X}, \mathbf{Y})] \leq \Big(C \cdot d \log D + d\Big)\, \mathcal{C}(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)).$$

Proof:

To write the pyramid match in terms of counts of *unmatched* points in the implicit partial matching, we define the *directed distance* between two histograms formed from point sets $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathcal{D}\left(H(\mathbf{X}), H(\mathbf{Y})\right) = \sum_{j=1}^{r} \mathcal{D}_j\left(H(\mathbf{X})^{(j)}, H(\mathbf{Y})^{(j)}\right), \text{ where} \tag{3.18}$$

$$\mathcal{D}_j(a, b) = \begin{cases} a - b, & \text{if } a > b \\ 0, & \text{otherwise} \end{cases} \tag{3.19}$$

The directed distance $\mathcal{D}(H_i(\mathbf{X}), H_i(\mathbf{Y}))$ counts the number of unmatched points at reso-

lution level i. Note that $\mathcal{D}\left(H_{-1}(\mathbf{X}), H_{-1}(\mathbf{Y})\right) = |\mathbf{X}|$ and $\mathcal{D}\left(H_{L-1}(\mathbf{X}), H_{L-1}(\mathbf{Y})\right) = 0$ by definition. The directed distance value decreases with i, as bins are increasing in size and more implicit matches are made. The change in the directed distance across levels is the decrease in the count of unmatched points from one level to the next, and therefore also serves to count the number of new matches formed at level i:

$$\mathcal{I}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) - \mathcal{I}\left(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})\right) = \mathcal{D}(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})) - \mathcal{D}(H_i(\mathbf{X}), H_i(\mathbf{Y})). \tag{3.20}$$

In terms of the directed distance, the pyramid match cost is

$$\begin{aligned}
\mathcal{P}_\Delta\left(\mathbf{X}, \mathbf{Y}\right) &= \sum_{i=0}^{L-1} d2^i \Big(\mathcal{D}\left(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y})\right) - \mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) \Big) \\
&= d\mathcal{D}\left(H_{-1}(\mathbf{X}), H_{-1}(\mathbf{Y})\right) + \sum_{i=0}^{L-2} d2^i \ \mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right) \\
&= d|\mathbf{X}| + \sum_{i=0}^{L-2} d2^i \ \mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right).
\end{aligned} \tag{3.21}$$

The expected value of the pyramid match cost is then

$$E\left[\mathcal{P}_\Delta\left(\mathbf{X}, \mathbf{Y}\right)\right] = d|\mathbf{X}| + \sum_{i=0}^{L-2} d2^i \ E\left[\mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right)\right]. \tag{3.22}$$

The optimal partial matching $\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*) = \left\{(\mathbf{x}_1, \mathbf{y}_{\pi_1^*}), \ldots, (\mathbf{x}_m, \mathbf{y}_{\pi_m^*})\right\}$ implies a graph whose nodes are those points in the input sets $\mathbf{X}$ and $\mathbf{Y}$ that participate in the matching (i.e., all points in $\mathbf{X}$ and a subset from $\mathbf{Y}$), and whose edges connect the matched pairs $(\mathbf{x}_i, \mathbf{y}_{\pi_i})$, for $1 \leq i \leq |\mathbf{X}|$. Let n_j be the number of edges in this graph that have lengths in the range $[d2^{j-1}, d2^j)$. A bound on the optimal partial matching may then be expressed as:

$$\begin{aligned}
\sum_j n_j \ d2^{j-1} &\leq \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right), \text{ or} \\
\sum_j n_j \ d2^j &\leq 2 \ \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right)
\end{aligned} \tag{3.23}$$

since for every j, all n_j edges must be of length at least $d2^{j-1}$.

The directed distance $\mathcal{D}$ counts at a given resolution how many points from set $\mathbf{X}$ are unmatched. Any edge in the optimal matching graph that is left "uncut" by the bin boundaries in histogram H_i contributes nothing to $\mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right)$. Any edge that is "cut" by the histogram contributes at most 1 to $\mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right)$. Therefore, the expected value of the directed distance at a given resolution is bounded by:

$$E\left[\mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right)\right] \leq \sum_j E\left[T_{ij}\right], \tag{3.24}$$

where T_{ij} is the number of edges in the optimal matching having lengths in the range $[d2^{j-1}, d2^j)$ that are cut by H_i.

The probability $\Pr(\text{cut}(\mathbf{x}, \mathbf{y}); i)$ that an edge for $(\mathbf{x}, \mathbf{y})$ in the optimal matching is cut by the histogram grid H_i is bounded by $\frac{||\mathbf{x}-\mathbf{y}||_1}{2^i}$. This can be seen by considering the probability of a bin boundary cutting an edge in any one of the d dimensions, taking into account that the bin boundaries are shifted independently in each dimension. An edge with length $||\mathbf{x} - \mathbf{y}||_1 > d2^i$ is certainly cut by the histogram grid at H_i. An edge with length $||\mathbf{x} - \mathbf{y}||_1 \leq d2^i$ may be cut in any dimension. In this case, the probability of a cut in dimension k for points $\mathbf{x} = [x_1, \ldots, x_d], \mathbf{y} = [y_1, \ldots, y_d]$ is $\frac{|x_k - y_k|}{2^i}$, for $1 \leq k \leq d$. The probability of a cut from H_i occurring in some dimension is then bounded by the sum of these probabilities:

$$\begin{aligned} \Pr(\text{cut}(\mathbf{x}, \mathbf{y}); i) &\leq \sum_{k=1}^{d} \frac{|x_k - y_k|}{2^i} \\ &\leq \frac{||\mathbf{x} - \mathbf{y}||_1}{2^i}. \end{aligned} \tag{3.25}$$

This provides an upper bound on the expected number of edges in the optimal matching that are cut at level i:

$$E\left[T_{ij}\right] \leq n_j \ \frac{d2^j}{2^i}, \tag{3.26}$$

since by definition $d2^j$ is the maximum distance between any pair of matched points counted by n_j.

Now with Eqns. 3.23 and 3.24 we have a bound for the expected directed distance at a certain resolution level:

$$E\left[\mathcal{D}\left(H_i(\mathbf{X}), H_i(\mathbf{Y})\right)\right] \leq \frac{1}{2^i} \ \sum_j n_j \ d2^j \leq \frac{1}{2^i} \ 2 \ \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right). \tag{3.27}$$

Relating this to the expected value of the pyramid match cost in Eqn. 3.22, we have

$$\begin{aligned} E\left[\mathcal{P}_\Delta\left(\mathbf{X}, \mathbf{Y}\right)\right] &\leq d|\mathbf{X}| + \sum_{i=0}^{L-2} d2^i \left(\frac{1}{2^i} \ 2 \ \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right)\right) \\ &\leq d|\mathbf{X}| + 2d \ (L-1) \ \mathcal{C}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)\right) \\ &\leq d|\mathbf{X}| + 2d \log D \ \mathcal{C}(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*)) \end{aligned} \tag{3.28}$$

since $L = \lceil \log_2 D \rceil + 1$.

As described in Section 3.1, points in sets $\mathbf{X}$ and $\mathbf{Y}$ are comprised of integer entries, insuring that the minimum inter-feature distance between unique points is 1. We also know that for the sake of measuring matching cost, $\mathbf{X}$ and $\mathbf{Y}$ are disjoint sets: $\mathbf{X} \cap \mathbf{Y} = \emptyset$; if there are any identical points in the two sets, we can consider them as discarded (in pairs) to produce strictly disjoint sets, since identical points contribute nothing to the matching cost. Since $\mathbf{X}, \mathbf{Y} \subseteq [D]^d$ and $\mathbf{X} \cap \mathbf{Y} = \emptyset$, we have $|\mathbf{X}| \leq \mathcal{C}(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi^*))$. That is, each edge in the optimal matching must have a length of at least 1, making the number of points in the

smaller set a lower bound on the cost of the optimal matching for any two sets. With this bound and Eqn. 3.28, we have the following bound on the expected pyramid match cost error

$$\begin{aligned} E\left[\mathcal{P}_\Delta\left(\mathbf{X},\mathbf{Y}\right)\right] &\leq d\,\mathcal{C}(\mathcal{M}(\mathbf{X},\mathbf{Y};\pi^*)) + 2d\log D\;\mathcal{C}(\mathcal{M}(\mathbf{X},\mathbf{Y};\pi^*)) \\ &\leq \Big(2d\log D + d\Big)\;\mathcal{C}(\mathcal{M}(\mathbf{X},\mathbf{Y};\pi^*)) \\ &\leq \Big(C\cdot d\log D + d\Big)\;\mathcal{C}(\mathcal{M}(\mathbf{X},\mathbf{Y};\pi^*)). \end{aligned} \tag{3.29}$$

In Chapter 6 I will also empirically evaluate the pyramid match error under various conditions, and I show that it is in practice significantly lower than that guaranteed by this bound. In Chapter 8 I provide results using the pyramid match as a kernel to perform several supervised learning tasks with images, and in Chapter 9 I discuss an unsupervised category learning approach based on the matching.

Chapter 4

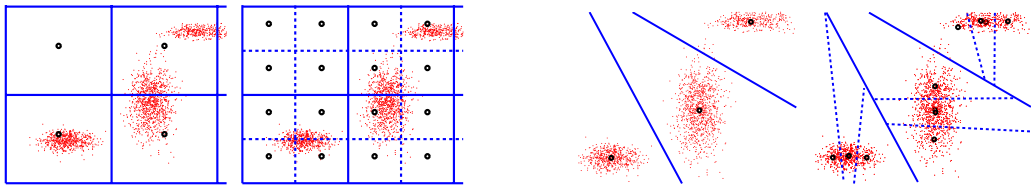
A Vocabulary-Guided Pyramid Match

In this chapter I give an extension of the pyramid match algorithm that specializes pyramids according to the distribution of features that appear in the input sets. As shown in the previous chapter, when using histograms with uniformly-shaped bins, the expected approximation error bound increases with the feature dimension. Here I introduce the idea of a *vocabulary-guided pyramid match*, which can in practice remain accurate even for sets with high-dimensional feature vectors, and benefits from taking advantage of the underlying structure of the feature space. Results pertaining to vocabulary-guided pyramid matching will be given in Chapters 6 and 8.

The main idea is to derive a hierarchical, data-dependent decomposition of the feature space that can be used to encode feature sets as multi-resolution histograms with irregularly-shaped bins. The given feature space is partitioned into a pyramid of non-uniformly shaped regions based on the distribution of a provided corpus of feature vectors (see Figure 4-1). Point sets are then encoded as multi-resolution histograms determined by that pyramid, and a weighted per-node intersection between any two histogram pyramids yields the approximate matching score for the original sets. The implicit matching estimates the inter-feature distances based on either their bin's diameter or their respective distances to the bin center.

I call this variant of the algorithm the vocabulary-guided pyramid match because the histogram pyramids are defined by the "vocabulary" or structure of the feature space. This extension makes the pyramid matching approach more generally applicable to situations where high-dimensional features are necessary or may improve performance. As my experiments in Chapter 6 will show, the vocabulary-guided pyramid match in practice can provide a good approximation to the optimal partial matching—even for high-dimensional feature spaces—but it is orders of magnitude faster to compute.

The idea of partitioning a feature space with vector quantization (VQ) is fairly widely used in practice; in the vision literature in particular, VQ has been used to establish a vocabulary of prototypical image features, from the "textons" of [70], to the "visual words" of [102]. More recently, Nister and Stewenius have shown that a hierarchical quantization of image



(a) Uniform bins (b) Vocabulary-guided bins

Figure 4-1: Rather than carve the feature space into uniformly-shaped partitions (left), for a vocabulary-guided pyramid match, the vocabulary (structure) of the feature space determines the partitions (right). As a result, the bins are concentrated on decomposing the space where features cluster, particularly for high-dimensional features. These figures depict the grid boundaries for two resolution levels for a 2-D feature space. In both (a) and (b), the left plot contains the coarser resolution level, and the right plot contains the finer one. Features are small points in red, bin centers are larger black points, and blue lines denote bin boundaries. The vocabulary-guided bins are irregularly shaped Voronoi cells.

features provides a scalable means of indexing into a very large feature vocabulary [85]. The actual tree structure employed is similar to the one constructed here; however, whereas the authors of [85] are interested in matching single features independently to one another to access an inverted file, my approach computes approximate correspondences between *sets* of features. Note the similar distinction between the problem I am addressing—approximate correspondence matching between sets—and the problem of efficiently identifying approximate or exact nearest-neighbor feature vectors (e.g., via k-d trees): in the former, the goal is a one-to-one correspondence between sets of vectors, whereas in the latter, a single vector is independently matched to a nearby vector.

4.1 Building Vocabulary-Guided Pyramids

The first step is to generate the structure of the vocabulary-guided pyramid to define the bin placement for the multi-resolution histograms used in the matching. This is a one-time process performed before any matching takes place. We would like the bins in the pyramid to follow the feature distribution and concentrate partitions where the features actually fall. To accomplish this, we perform hierarchical clustering on a sample of representative feature vectors from the feature space F.

We randomly select some example feature vectors from the feature type of interest to form the representative feature corpus, and perform hierarchical k-means clustering with the Euclidean distance to build the pyramid tree. Other hierarchical clustering techniques, such as agglomerative clustering, are also possible and do not change the operation of the method. For this unsupervised clustering process there are two parameters: the number of levels in the tree L, and the branching factor k.[1] The initial corpus of features is clustered

[1]For the pyramid match based on uniform bins described in Chapter 3, L was computed directly from the feature space aspect ratio D. For the vocabulary-guided bins, L still reflects the number of resolution levels, but it is a user-supplied parameter.

into k top-level groups, where group membership is determined by the Voronoi partitioning of the feature corpus according to the k cluster centers. Then the clustering is repeated recursively $L-1$ times on each of these groups, filling out a tree with L total levels containing k^i bins (nodes) at level i, where levels are counted from the children of the root ($i = 0$) to the leaves ($i = L-1$).[2] The bins are irregularly shaped and sized, and their boundaries are determined by the Voronoi cells surrounding the cluster centers (see Figure 4-1). For each bin in the vocabulary-guided pyramid we record its diameter—the maximal inter-feature distance between any points from the initial feature corpus that were assigned to it.

Once we have constructed a vocabulary-guided pyramid, we can embed point sets from S as multi-resolution histograms. A point's placement in the histogram pyramid is determined by comparing it to the appropriate k bin centers at each of the L pyramid levels. The histogram count is incremented for the bin (among the k choices) that the point is nearest to in terms of the same distance function used to cluster the initial corpus. We then push the point down the tree and continue to increment finer level counts only along the branch (bin center) that is chosen at each level. So a point is first assigned to one of the top-level clusters, then it is assigned to one of *its* children, and so on recursively. This amounts to a total of kL distances that must be computed between a point and the pyramid's bin centers, and this is the primary cost increase over the pyramid match with uniform bins.

It is feasible with the vocabulary-guided pyramid to apply a "soft" assignment function to select multiple bins into which each point is entered. Doing so may potentially alleviate quantization effects that are possible with a hard-assignment clustering. However, the more neighboring bins that are incorporated into each assignment, the more the sparseness of the resulting pyramids is reduced, which will influence the cost of pyramid intersections.

Given the bin structure of the vocabulary-guided pyramid, a point set $\mathbf{X}$ is mapped to its pyramid $\Psi_v(\mathbf{X})$:

$$\begin{aligned} \Psi_v(\mathbf{X}) &= [H_0(\mathbf{X}), \ldots, H_{L-1}(\mathbf{X})] \text{ , with} \\ H_i(\mathbf{X}) &= [\langle \mathbf{p}, n, d \rangle_1, \ldots, \langle \mathbf{p}, n, d \rangle_{k^i}], \end{aligned} \tag{4.1}$$

where $H_i(\mathbf{X})$ is a k^i-dimensional histogram associated with level i in the vocabulary-guided pyramid, $\mathbf{p} \in \mathbb{Z}^i$ for entries in $H_i(\mathbf{X})$, and $0 \leq i < L$. Each entry in this histogram is a triple $\langle \mathbf{p}, n, d \rangle$ giving the bin index, the bin count, and the bin's points' maximal distance to the bin center, respectively. Note the small change in notation: Ψ referred to a pyramid map with uniform bins in Chapter 3, while here Ψ_v refers to a vocabulary-guided pyramid map with non-uniform bins.

Storing the vocabulary-guided pyramid itself requires space for $O(k^L)$ d-dimensional feature vectors, i.e., all of the cluster centers. However, as with the pyramids with uniform bins, each point set's histogram embedding is stored sparsely, meaning only $O(mL)$ nonzero bin counts are maintained to encode the entire pyramid for a set with m features. This is an important point: we do not store $O(k^L)$ counts for every point set; $H_i(\mathbf{X})$ is represented by at most m triples having $n > 0$.

[2] For notation convenience, for the vocabulary-guided pyramids, level L refers to the finest resolution, while in the previous chapter the uniform bins at level L were the coarsest.

We achieve a sparse implementation as follows: each vector in a set is pushed through the tree as described above. At every level i, we record a $\langle \mathbf{p}, n, d \rangle$ triple describing the nonzero entry for the current bin. The vector $\mathbf{p} = [p_1, \ldots, p_i]$, $p_j \in [1, k]$, for $1 \leq j \leq i$ denotes the indices of the clusters traversed from the root so far, $n \in \mathbb{Z}^+$ denotes the count for the bin (initially 1), and $d \in \Re$ denotes the distance computed between the inserted point and the current bin's center. Upon reaching the leaf level, $\mathbf{p}$ is an $L-1$-dimensional path-vector indicating which of the k bins were chosen at each level after the root, and every path-vector uniquely identifies some bin on the pyramid.

Initially, an input set with m features yields a total of mL such triples—there is one nonzero entry per level per point, and each has $n = 1$. Then each of the L lists of entries is sorted by the index vectors ($\mathbf{p}$ in the triple), and they are collapsed to a list of sorted nonzero entries with unique indices: when two or more entries with the same index are found, they are replaced with a single entry with the same index for $\mathbf{p}$, the summed counts for n, and the maximum distance for d. The sorting is done in linear time using integer sorting algorithms. Maintaining the maximum distance of any point in a bin to the bin center will allow us to efficiently estimate inter-point distances at the time of matching, as I will describe in the next section.

4.2 Computing a Vocabulary-Guided Pyramid Match

Given two point sets' vocabulary-guided pyramid encodings, we efficiently compute the pyramid match score. The vocabulary-guided pyramid will provide a partitioning of the feature space at multiple resolutions that is used to direct the matching. As before, the basic intuition is to start collecting groups of matched points from the bottom of the tree up, i.e., from within increasingly larger partitions. In this way, we will first consider matching the closest points (at the leaves), and as we climb to the higher-level clusters in the pyramid we will allow increasingly further points to be matched. We define the number of *new* matches within a bin to be a count of the minimum number of points either of the two input sets contributes to that bin, minus the number of matches already counted by any of its child bins. A weighted sum of these counts yields an approximate matching score.

Note in the following that aside from implementation details surrounding the sparse tree construction, the primary distinction for the matching process here as opposed to the matching described in Chapter 3 is that now match counts must be made per *node*, rather than per level, and it matters which bin is a child of which. That is because every node at a given level has a unique diameter and so matches accumulated in each node must be weighted accordingly.

Let $n_{ij}(\mathbf{X})$ denote the element n from $\langle \mathbf{p}, n, d \rangle_j$, the j^{th} bin entry of histogram $H_i(\mathbf{X})$, and let $c_h\left(n_{ij}(\mathbf{X})\right)$ denote the element n for the h^{th} child bin of that entry, $1 \leq h \leq k$. Similarly, let $d_{ij}(\mathbf{X})$ refer to the element d from the same triple. Given point sets $\mathbf{X}$ and

$\mathbf{Y}$, we compute the matching score via their pyramids $\Psi_v(\mathbf{X})$ and $\Psi_v(\mathbf{Y})$ as follows:

$$\tilde{\mathcal{P}}_\Delta\left(\Psi_v(\mathbf{X}), \Psi_v(\mathbf{Y})\right) = \sum_{i=0}^{L-1} \sum_{j=1}^{k^i} w_{ij} \left[\min\left(n_{ij}(\mathbf{X}), n_{ij}(\mathbf{Y})\right) - \sum_{h=1}^{k} \min\left(c_h\left(n_{ij}(\mathbf{X})\right), c_h\left(n_{ij}(\mathbf{Y})\right)\right) \right] \tag{4.2}$$

The outer sum loops over the levels in the pyramids; the second sum loops over the bins at a given level, and the innermost sum loops over the children of a given bin. The first min term reflects the number of matchable points in the current bin, and the second min term tallies the number of matches already counted at finer resolutions (in child bins). Note that as the leaf nodes have no children, when $i = L - 1$ the last sum is zero. All matches are new at the leaves. The matching scores are normalized according to the size of the input sets in order to not bias towards (for similarity) or against (for cost) larger sets.

The number of new matches calculated for bin j at level i is weighted by w_{ij}, an estimate of the distance between points contained in the bin. As in Chapter 3, to use the matching as a cost function, weights are set as the distance estimates; to use as a similarity measure, weights are set as (some function of) the inverse of the distance estimates.

With a vocabulary-guided pyramid match there are two alternatives for the distance estimate, which are shown below for a cost measure. The first option (a) uses weights based on the diameters of the pyramid's bins; option (b) uses input-dependent weights based on the maximal distances of the points in the bin to its center:

$$\begin{aligned} (a) \quad w_i &= A_{ij} \\ (b) \quad w_i &= d_{ij}(\mathbf{X}) + d_{ij}(\mathbf{Y}) \end{aligned} \tag{4.3}$$

where A_{ij} is the diameter of the j^{th} bin of the vocabulary-guided pyramid's i^{th} level.

The first option is a conservative estimate of the actual inter-point distances in the bin if the corpus of features that built the pyramid is representative of the feature space. As a similarity measure, this choice has the advantage of providing a guaranteed Mercer kernel (see below), and it also eliminates the need to store a distance d in the entry triples. This weighting scheme can be viewed as analogous to the maximal-diagonal weighting scheme employed in Chapter 3 for the uniformly-sized hypercube bins.

On the other hand, we can also produce input-specific distance estimates. The second option (b) is to estimate the distance between any two points in the bin as the sum of the stored maximal to-center distances from either input set. In fact, since points are inserted into bins under a metric measure, due to the triangle inequality this is a true upper bound on the furthest any two points could be from one another. Note that this second weighting option exploits the distances to the bin centers that were already computed to insert the points into the histograms; maintaining these distances costs space but not additional time. This weighting has the potential to provide tighter estimates of inter-feature distances, which I confirm experimentally in Chapter 6; however, we do not have a proof to guarantee this weighting will yield a Mercer kernel.

Just as the pyramids are encoded sparsely, I derive a means to compute intersections in

Eqn. 4.2 without ever traversing the entire pyramid tree. Given two sparse lists $H_i(\mathbf{X})$ and $H_i(\mathbf{Y})$ which have been sorted according to the bin indices, we obtain the minimum counts in linear time in the number of nonzero entries by moving pointers down the lists and processing only entries that share an index, making the time required to compute an approximate matching between two pyramids $O(mL)$. The only inter-feature distances computed are the kL distances need to insert a point into the pyramid, and this small one-time cost is amortized every time we re-use a pyramid embedding to approximate another matching against a different point set.

The proof of the expected error bounds of the pyramid match given in Section 3.5 of the previous chapter applies only to the pyramid match when using randomly shifted histograms with uniform bins. That proof relied on the fact that the bin boundaries were placed at random in the feature space, allowing us to measure the probability that an edge connecting two points would be cut by a boundary. I do not have a similar bound here for the vocabulary-guided pyramid match, as its bin boundaries are placed according to the distribution of features in a supplied corpus. However, as I will show in Chapter 6, in practice the vocabulary-guided pyramid match remains accurate and efficient even for high-dimensional feature spaces, while pyramid matching with uniform bins is limited in practice to lower-dimensional features (with d on the order of 10). It is possible that theoretical error bounds could be shown for the vocabulary-guided variant based on knowledge of the intrinsic dimensionality of the feature space, or by fitting some parametric form over the features.

For the increased accuracy, there are some complexity trade-offs. The pyramid match with uniform bins does not require computing any distances to place the points into bins; their uniform shape and size allows points to be placed directly via division by bin size. On the other hand, sorting the bin indices with the vocabulary-guided method usually has a lower complexity, since the integer values only range to k, the branching factor, which will typically be much smaller than the feature aspect ratio D that bounds the range in the uniform bin case. In addition, as I will show in Chapter 6, in high dimensions the cost of extracting an explicit correspondence field using the pyramids with uniform bins approaches the cubic cost of the optimal measure, whereas it can remain linear with the vocabulary-guided pyramids, assuming features are not uniformly distributed.

4.3 The Vocabulary-Guided Pyramid Match as a Mercer Kernel

The vocabulary-guided pyramid match forms a Mercer kernel. We can rewrite Eqn. 4.2 as:

$$\tilde{\mathcal{P}}_{\Delta}\left(\Psi_v(\mathbf{X}), \Psi_v(\mathbf{Y})\right) = \sum_{i=0}^{L-1} \sum_{j=1}^{k^i} \left(w_{ij} - p_{ij}\right) \min\left(n_{ij}(\mathbf{X}), n_{ij}(\mathbf{Y})\right), \tag{4.4}$$

where p_{ij} refers to the weight associated with the parent bin of the j^{th} node at level i.

Since the intersection (min) operation is positive-definite [86], and since kernels are closed under summation and scaling by a positive constant [99], we have that the vocabulary-

guided pyramid match is a Mercer kernel if $w_{ij} \geq p_{ij}$. This inequality holds if every child bin receives a similarity weight that is greater than its parent bin, or rather that every child bin has a distance estimate that is less than that of its parent. Indeed this is the case for our first weighting option (Eqn. 4.3 (a)), where w_{ij} is set to be inversely proportional to the diameter of the bin in the vocabulary-guided pyramid. It holds by definition of the hierarchical clustering: the diameter of a subset of points must be less than or equal to the diameter of *all* those points. We cannot make this guarantee for the second weighting option.

In Chapter 6 I demonstrate the accuracy advantages of the vocabulary-guided pyramid match for high-dimensional features, and in Chapter 8 I report object recognition results using the vocabulary-guided pyramid match as a kernel.

Chapter 5

Approximate Similarity Search with Approximate Matchings

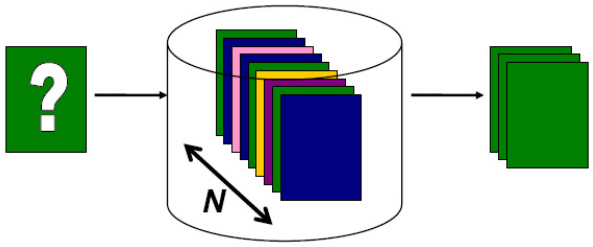
Even when there is an efficient way to compare two inputs, there is a second computational hurdle: given a query example, how can a very large database of examples be efficiently indexed to retrieve the most similar items? A naive linear scan of all database items quickly becomes intractable for large databases. Considering that web image databases and even personal photo collections can easily exceed thousands to millions of images, the ability to efficiently index under a similarity measure of choice is crucial for real-world applications.

For similarity search in the d-dimensional Euclidean space, structures such as a k-d tree may be pre-computed to reduce the searching process to a smaller portion of the entire database and avoid a brute-force linear scan. However, these techniques assume inputs are in a vector space, and in practice break down for high-dimensional inputs.

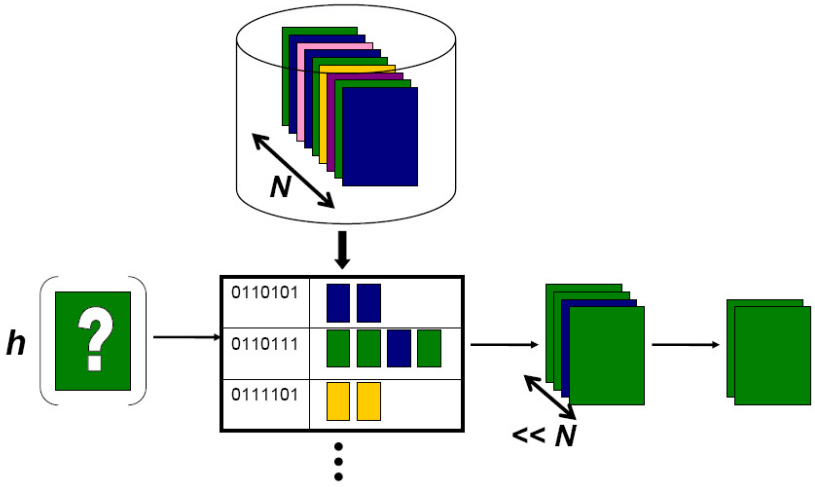
In this chapter I describe how to perform approximate similarity search where inputs are sets of vectors and similarity is measured by the matching between two sets. To do this for sets with equal cardinality, I make use of an existing randomized hashing technique called Locality-Sensitive Hashing [56, 38] and the embedding of Indyk and Thaper [57]. This framework is reviewed in Section 5.1. Then in Section 5.2, I derive an embedding based on the pyramid match to allow randomized hashing over the matching between variable-sized sets. These two hashing approaches allow us to perform database retrievals in sub-linear time in the number of database examples. Related results are provided in Chapter 7.

5.1 Locality-Sensitive Hashing for Equal-Sized Sets

Locality-Sensitive Hashing (LSH) is a randomized hashing algorithm developed by Indyk and Motwani [56]. LSH indexes a database of examples residing in a normed space by a number of hash tables, such that the probability of collision is high for similar examples and low for dissimilar ones. In particular, LSH guarantees that if for a query point $\mathbf{q}$ there exists a point in the database $\mathbf{p}$ such that $\mathcal{D}(\mathbf{p}, \mathbf{q}) \leq r$, then (with high probability) a point $\mathbf{p}'$ is



(a) Database query via a naive linear scan



(b) Sub-linear time database query via randomized hashing

Figure 5-1: For large databases, a naive linear scan to compute similarities against a query (a) quickly becomes impractical, even with the most efficient similarity measure. With randomized hashing techniques such as Locality-Sensitive Hashing, we can perform sub-linear time queries of very large databases (b). The chosen hash function h must guarantee that examples similar under the measure of interest will collide in the hash buckets. All N database examples are hashed into tables where there will be a high probability that similar examples collide. Then given a new query example, only those database entries that share a hash bucket with the query must be exhaustively searched to identify the most similar examples.

returned such that $\mathcal{D}(\mathbf{p}', \mathbf{q}) \leq (1+\epsilon)r$. Otherwise, the absence of such a point is reported. The query time for a database of N d-dimensional examples is bounded by $O(dN^{1/(1+\epsilon)})$ (see Figure 5-1).

Locality-Sensitive Hashing presumes inputs are in a normed space. To hash over a specialized similarity or distance function of interest, an appropriate family of hash functions must be selected; or, the hash functions proposed in [38, 56] may be applied if the inputs can be embedded in such a way that a norm (L_1 or L_2) over the embeddings induces that similarity function.

As described in Chapter 2, Indyk and Thaper have previously shown that the L_1 distance between randomly shifted multi-resolution histograms provides a low-distortion embedding for the least-cost matching between equal-mass point sets [57]. The histograms provide an embedding of the original data into a normed space, making it possible to apply LSH. The distortion C of a metric embedding E describes how much information loss the embedding induces: $\mathcal{C}(\mathcal{M}(\mathbf{A}, \mathbf{B}; \pi^*)) \leq ||E(\mathbf{A}) - E(\mathbf{B})||_{L_1} \leq C \cdot \mathcal{C}(\mathcal{M}(\mathbf{A}, \mathbf{B}; \pi^*))$. The distortion of the embedding given by Indyk and Thaper has an upper bound of $O(d \log D)$ for feature aspect ratio D and dimension d.[1] In [19], Charikar has independently explored hash functions for matching with equal-sized sets that have closely related properties to those proposed in [38, 57].

In [57] the authors show an image retrieval application using the L_1 embedding. Global color histograms (or "signatures" over the color space) are extracted for each image as in Rubner et al. [92], and a database of these color histograms is embedded via the multi-resolution histogram. Then given a query image, sub-linear nearest-neighbor search with LSH is performed to retrieve the most relevant items in terms of the matching between the color signatures.

I have built upon this framework to show Locality-Sensitive Hashing over the approximate matching between sets of *local* image feature vectors. The method proceeds as follows: local shape or appearance features are extracted from a database of images, and each image's features are treated as a uniformly weighted point set. Note that for this approach, an equal number of features must be sampled from every image. Using the L_1 embedding over the point sets, one sparse vector is produced for each input shape or image. Next, a set of random LSH hash functions are generated, and all of the embedded database vectors are entered into the hash tables. Both the database embedding and hash table construction is performed off-line.

Then, given a novel image example, the embedding for its features is computed using the same random grid translations used for the database embedding. Finally, images similar to the novel query are retrieved from the database by computing the L_1 distance between the query's embedding and only those vectors in the union of the hash buckets that are indexed by the query's embedding (see Procedures 1 and 2). LSH guarantees that we will only have to exhaustively compare the query to $O(N^{1/(1+\epsilon)})$ embedding vectors from the database; with a typical setting of $\epsilon = 1$, this means that only $O(\sqrt{N})$ items are searched.

[1] While the embedding in [57] is described as an approximation for the Earth Mover's Distance (EMD), it assumes inputs are weighted point sets with total equal mass, and so it does not produce partial matchings as the original EMD measure developed by Rubner et al. does [92].

Procedure 1 To prepare an image data set for hashing:

Given: A data set of N images $\{I_1, \ldots, I_N\}$, random LSH functions $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_t]$, randomly translated histogram grids H_l, each with side lengths 2^l, $l = 0, \ldots, L$, $L = \lceil \log D \rceil$, and neighborhood radius r:

1: **for all** $i = 1, \ldots, N$ **do**
2: Detect in image I_i all interest points $\{\mathbf{p}_1, \ldots, \mathbf{p}_{n_i}\}$, either with a chosen interest operator or simply on a uniform grid.
3: Extract a descriptor $\mathbf{x}_j$ having the desired invariance properties (e.g., scale- or rotation-invariance) from the image patch or region centered at each $\mathbf{p}_j$ to form the set of features $\mathbf{X}_i = \{\mathbf{x}^{\mathbf{p}_1}, \ldots, \mathbf{x}^{\mathbf{p}_{n_i}}\}$.
4: Weight each feature $\mathbf{x}^{\mathbf{p}_j}$ by $\frac{1}{n_i}$ and apply the L_1 embedding: $E(\mathbf{X}_i) = \left[H_0(\mathbf{X}_i), \ldots, 2^l H_l(\mathbf{X}_i), \ldots, DH_{L-1}(\mathbf{X}_i)\right]$, producing one sparse vector.
5: Insert vector $E(\mathbf{X}_i)$ into hash tables $\mathbf{H}$, and record its hash buckets $[b_1, \ldots, b_t]$.
6: **end for**

Procedure 2 To query the prepared data set to find similar images:

Given: Image I_q with set of features $\mathbf{X}_q$, embedding $E(\mathbf{X}_q)$, and random LSH functions $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_t]$:

1: Hash into $\mathbf{H}$ with $E(\mathbf{X}_q)$, yielding hash bucket indices $[b_1, \ldots, b_t]$
2: **for all** $c = 1, \ldots, t$ **do**
3: Compute the L_1 distance between $E(\mathbf{X}_q)$ and the W database embedding vectors $\{E(\mathbf{X}^1), \ldots, E(\mathbf{X}^W)\}_c$ that share bucket b_c, $W << N$.
4: **end for**
5: Sort $\cup_{c=1}^{t} \left\{E(\mathbf{X}^1), \ldots, E(\mathbf{X}^W)\right\}_c$ according to their L_1 distance to $E(\mathbf{X}_q)$ to obtain a ranked image list that includes the approximate r-neighbors of I_q.

In Chapter 7 I demonstrate this L_1 embedding and LSH applied to contour shape retrievals for multiple large databases.

5.2 Pyramid Match Hashing for Variable-Sized Sets

The above hashing technique assumes that inputs are sets with equal numbers of features, or weighted sets with equal total mass. In this work I am also interested in being able to match sets of variable sizes, since it is highly unlikely that every image will have a local representation comprised of the same number of parts. That is, we would like to be able to apply randomized hashing over sets that are compared by a partial matching as approximated by the pyramid match (see Chapter 3).

To hash with sets of variable sizes, below I derive a new embedding via the pyramid match for approximate similarity search under the matching similarity normalized by the sizes of both input sets. I then show that locality sensitive hashing is not possible for a complete partial match, where no penalty at all is accumulated for unmatched points.

5.2.1 Random Hyperplane Hash Functions for the Pyramid Match

In [19], Charikar gives a slightly different definition of Locality-Sensitive Hashing than the one in [38] described above, although it is similar in spirit. In this case, a locality sensitive hashing scheme is a distribution on a family $\mathcal{F}$ of hash functions h operating on a collection of objects, such that for two objects x, y,

$$\Pr_{h \in \mathcal{F}}[h(x) = h(y)] = sim(x, y), \tag{5.1}$$

where $sim(x, y)$ is some similarity function defined on the collection of objects [19]. In other words, the probability that two inputs collide in the hash table is equal to the similarity between them, and so highly similar objects will be indexed by the hash table with high probability.

In [39], Goemans and Williamson provide a randomized algorithm for the MAX-CUT problem using semidefinite programming. As part of this work, they prove that given a collection of vectors $\{\vec{v}_1, \ldots, \vec{v}_n\}$ belonging to the unit sphere S_n, and a randomly generated vector $\vec{r}$, the probability that any two vectors $\vec{v}_i$ and $\vec{v}_j$ each have dot products with $\vec{r}$ having opposite signs is related to the vectors as follows:

$$\Pr[\text{sgn}(\vec{v}_i \cdot \vec{r}) \neq \text{sgn}(\vec{v}_j \cdot \vec{r})] = \frac{1}{\pi} \cos^{-1}(\vec{v}_i \cdot \vec{v}_j). \tag{5.2}$$

That is, the probability a random hyperplane separates two vectors is directly proportional to the angle $\cos^{-1}(\vec{v}_i \cdot \vec{v}_j)$.

This was exploited for locality sensitive hashing by Charikar [19], who considers a hash function family for set similarity to generate hash keys using the above property of random hyperplanes. Given a database of vectors in $\Re^d$, a vector $\vec{r}$ is chosen at random from the d-dimensional Gaussian distribution. The corresponding hash function $h_{\vec{r}}$ accepts a vector $\vec{u} \in \Re^d$, and is defined as:

$$h_{\vec{r}}(\vec{u}) = \begin{cases} 1, & \text{if } \vec{r} \cdot \vec{u} \geq 0 \\ 0, & \text{if } \vec{r} \cdot \vec{u} < 0 \end{cases}. \tag{5.3}$$

Then, drawing on the relationship in Eqn. 5.2, a valid locality sensitive hashing scheme is:

$$\begin{aligned} \Pr[h_{\vec{r}}(\vec{v}_i) = h_{\vec{r}}(\vec{v}_j)] &= 1 - \frac{\theta(\vec{v}_i, \vec{v}_j)}{\pi}, \text{ where} \\ \theta(\vec{v}_i, \vec{v}_j) &= \cos^{-1}\left(\frac{\vec{v}_i \cdot \vec{v}_j}{\sqrt{|\vec{v}_i| \ |\vec{v}_j|}}\right). \end{aligned} \tag{5.4}$$

Charikar points out that if the vectors $\vec{v}_i$ and $\vec{v}_j$ are treated as characteristic vectors for sets, this could serve as a set similarity measure.

In the following, I show that we can achieve hashing over the pyramid match kernel using this hash function family. We can consider the pyramid mapping $\Psi(\mathbf{X})$ as an embedding for a point set $\mathbf{X}$, and incorporate the weights used in the pyramid matching $\mathcal{P}_\Delta$. Then by applying a type of unary encoding, we will have an embedding for each point set that under a dot product yields the un-normalized pyramid match kernel value.

From Eqn. 3.16 in Chapter 3, we have the definition of the (un-normalized) pyramid match kernel:

$$\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right) = w_{L-1}\mathcal{I}\left(H_{L-1}(\mathbf{Y}), H_{L-1}(\mathbf{Z})\right) + \sum_{i=0}^{L-2}(w_i - w_{i+1})\,\mathcal{I}\left(H_i(\mathbf{Y}), H_i(\mathbf{Z})\right), \quad (5.5)$$

where w_i is the weight associated with level i of the pyramids, and the value of the weights decreases with i (e.g., $w_i = \frac{1}{d2^i}$ is a valid option).

Given a histogram H that contains r bins, and a weight w, let $[wH]$ denote an r-dimensional vector giving the counts in each bin of the histogram, with each count scaled by w. Note that this weighting is distributive over histogram intersection; that is, a weighted histogram intersection value is equivalent to the intersection of the weighted histograms, or $w\,\mathcal{I}\left(H(\mathbf{Y}), H(\mathbf{Z})\right) = \mathcal{I}\left([wH(\mathbf{Y})], [wH(\mathbf{Z})]\right)$.

Let $\mathcal{U}([wH])$ denote the following (padded) unary encoding of the histogram H weighted by w:

$$\mathcal{U}\left([wH]\right) = \left(\underbrace{\overbrace{1,\ldots,1}^{wH^{(1)}},\ \overbrace{0,\ldots,0}^{M-wH^{(1)}}}_{\text{first bin}}, \ldots, \underbrace{\overbrace{1,\ldots,1}^{wH^{(r)}},\ \overbrace{0,\ldots,0}^{M-wH^{(r)}}}_{\text{last bin}}\right), \quad (5.6)$$

where M is the maximum possible weighted count in any histogram bin, and $H^{(j)}$ refers to the count in bin j of histogram H.[2] Let $\mathbf{r}_i(\mathbf{X})$ refer to the histogram for set $\mathbf{X}$ at pyramid level i, weighted by $w = w_i - w_{i+1}$:

$$\mathbf{r}_i(\mathbf{X}) = \left[(w_i - w_{i+1})\ H_i(\mathbf{X})\right]. \quad (5.7)$$

The following embedding f serves to map the set of vectors $\mathbf{X}$ to a single vector:

$$f(\mathbf{X}) = \Big[\mathcal{U}(\mathbf{r}_0(\mathbf{X}))\ \mathcal{U}(\mathbf{r}_1(\mathbf{X}))\ \mathcal{U}(\mathbf{r}_2(\mathbf{X})), \ldots, \mathcal{U}(\mathbf{r}_{L-2}(\mathbf{X}))\ \mathcal{U}([w_{L-1}H_{L-1}(\mathbf{X})])\Big]. \quad (5.8)$$

The dot product between two such encodings for sets $\mathbf{Y}$ and $\mathbf{Z}$ yields the un-normalized pyramid match kernel score from Eqn. 5.5 above:

$$f(\mathbf{Y}) \cdot f(\mathbf{Z}) = \tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right). \quad (5.9)$$

The length $|f(\mathbf{Y})|$ of an encoding vector $f(\mathbf{Y})$ is simply the sum of its total number of nonzero (one) entries. Since self-intersection of a histogram returns the number of total points in the histogram ($\mathcal{I}(H(\mathbf{Y}), H(\mathbf{Y})) = |\mathbf{Y}|$), the length of an embedding vector is

[2] If weighted counts are real-valued, this process can in theory proceed by scaling to a given precision and truncating to integers. With the normalization factor also scaled, the output remains equivalent.

going to be equivalent to the original set's self-similarity score under the pyramid match:

$$\begin{aligned}
|f(\mathbf{Y})| &= w_{L-1}\,|\mathbf{Y}| + \sum_{i=0}^{L-2}(w_i - w_{i+1})\,|\mathbf{Y}| \\
&= w_{L-1}\mathcal{I}(H_{L-1}(\mathbf{Y}), H_{L-1}(\mathbf{Y})) + \sum_{i=0}^{L-2}(w_i - w_{i+1})\mathcal{I}(H_i(\mathbf{Y}), H_i(\mathbf{Y})) \\
&= \tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Y})\right). \qquad (5.10)
\end{aligned}$$

Putting these pieces together, we have an embedding of the pyramid match kernel that allows us to perform sub-linear time similarity search with random hyperplane hash functions. With the new embedding in Eqn. 5.8 and the hashing guarantee from Eqn. 5.4, we have:

$$\begin{aligned}
\Pr\left[h_{\vec{r}}(f(\mathbf{Y})) = h_{\vec{r}}(f(\mathbf{Z}))\right] &= 1 - \frac{\theta(f(\mathbf{Y}), f(\mathbf{Z}))}{\pi}, \text{ where} \\
\theta(f(\mathbf{Y}), f(\mathbf{Z})) &= \cos^{-1}\left(\frac{f(\mathbf{Y}) \cdot f(\mathbf{Z})}{\sqrt{|f(\mathbf{Y})|\;|f(\mathbf{Z})|}}\right) \\
&= \cos^{-1}\left(\frac{\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right)}{\sqrt{\tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Y})\right) \times \tilde{\mathcal{P}}_\Delta\left(\Psi(\mathbf{Z}), \Psi(\mathbf{Z})\right)}}\right) \\
&= \cos^{-1}\left(\mathcal{P}_\Delta\left(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})\right)\right). \qquad (5.11)
\end{aligned}$$

Notice that this last term is the normalized pyramid match kernel value, where normalization is done according to the product of the self-similarity scores (Eqn. 3.12 in Chapter 3). In Section 5.2.3 I discuss why locality sensitive hashing is not possible with the alternative normalization scheme, where scores are normalized by the minimum input cardinality.

We do not need to explicitly expand the components $\mathbf{r}_i(\mathbf{X})$ into their unary encodings. Likewise, we do not need to generate an entry for every dimension of the random vector $\vec{r}$ to compute a hash key from $f(\mathbf{X})$. Instead, the counts in $H_i(\mathbf{X})$ indicate which entries in $\vec{r}$ will result in a nonzero contribution to $f(\mathbf{X}) \cdot \vec{r}$, that is, those entries where the encoding for $\mathbf{r}_i(\mathbf{X})$ would be 1, not 0. The inner product between the random vector and the embedding is then the sum of those particular entries in $\vec{r}$, and the sign of this sum determines the hash key value $h_{\vec{r}}(f(\mathbf{X}))$.

A similar embedding is possible with the vocabulary-guided pyramid match, since the intersected vocabulary-guided pyramid can analogously be written as a dot product between weighted histograms (refer to Eqn. 4.4). The first weighting option (Eqn. 4.3 (a)) must be used in this case to have a set of "global" weights to compute the embedding that is independent of any two given input sets. As with the kernel or cost value computation, the vocabulary-guided pyramid match embedding is also computed at the level of the bins, not at the level of the pyramid resolutions, since each bin may have a unique size and shape and must be weighted accordingly.

5.2.2 Comparison of Embedding Choices

In Chapter 3 I presented two possible normalization schemes: normalization by the product of both sets' self-similarities, or normalization by only the minimum cardinality input set (see Eqns. 3.11 and 3.12). The hashing described above makes use of the former, which includes the sizes of *both* input sets. This yields a correspondence measure between two variable-sized sets that does include some penalty for the unmatched points in the larger set.

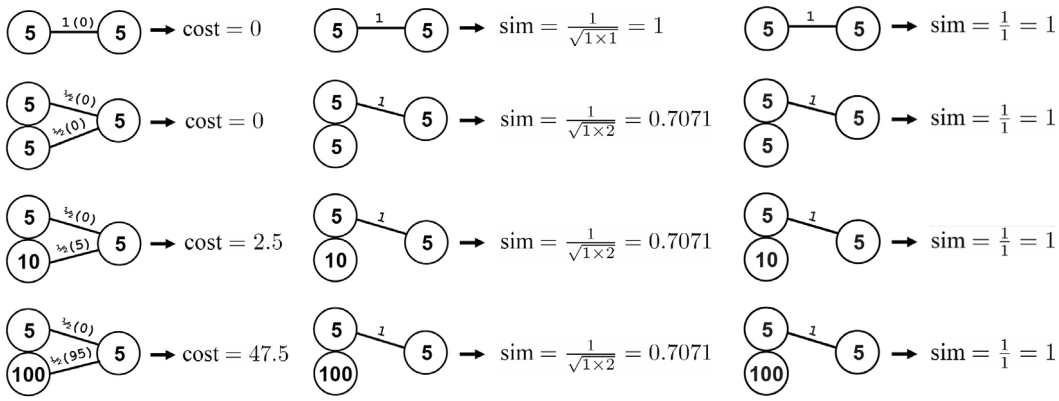
For example, consider two sets; with the minimum cardinality normalization, their pyramid match kernel value would remain constant if we were to add more and more features to one of the sets. In contrast, with the product normalization, the pyramid match kernel value would slowly decrease as we added those features. When is this a desired property for image matching? If there is expected to be an unknown amount of clutter, background, or unmatched features in both of the images being matched, this normalization is reasonable. The best matching will be the one that can find good matches for all the features in both sets. An image matching with more clutter (unmatchable features) will receive a lower similarity weight than an image matching with fewer unmatched features. However, note that pyramid match hashing will not care *how different* the unmatched features are to any features in the other set; that is, the penalty is only relative to *how many* unmatched features there are (see Figure 5-2). I verify this property experimentally in Chapter 6, Section 6.2.

The L_1 embedding of Indyk and Thaper is applicable only if the sum of the total weight attached to points in any input set is constant. One way to achieve equal weights for variable-sized sets is to normalize the feature weights to force an equal sum of weights for each set. However, doing so alters the actual distribution of features that were extracted from the image, forces a bijective matching, and thus alters the meaning of the matching computed between two sets. The hashing approach I have proposed does not require altering the input feature set; thus the true feature distribution is preserved, as is the meaning behind matching *parts* of feature sets to one another. Whether a normalized partial match or a bijective matching over re-weighted features is more appropriate will depend on the application and what is desired from the matching scores.

5.2.3 Existence of Locality-Sensitive Hash Functions for a Partial Match

The above hashing framework relies on a metric-case of a matching between variable-sized sets. In fact the metric property is necessary to allow locality sensitive hashing. For a metric space M under metric distance $\mathcal{D}$, the triangle inequality states that $\mathcal{D}(x,z) \leq \mathcal{D}(x,y) + \mathcal{D}(y,z)$, $\forall x,y,z \in M$. In [19] it is shown that for any similarity function that admits a locality sensitive hash function family as defined in Eqn. 5.1, the distance function $1 - sim(x,y)$ satisfies the triangle inequality.

Given this fact, I can show that there is no locality sensitive hash function family corresponding to the similarity of the partial matching over sets $\mathbf{X}$ and $\mathbf{Y}$ normalized by the minimum input set size. Let the partial match similarity judge similarity between points as being inversely proportional to their distance, as it is in the pyramid match. The corre-



(a) Costs with normalized feature weights to create equal-mass sets

(b) Similarities with partial match and normalization by both input cardinalities

(c) Similarities with partial match and normalization by minimum cardinality

Figure 5-2: These examples illustrate differences between using a bijective matching to compare variable-sized sets whose feature weights are normalized to sum to one (a), versus using a normalized partial matching to compare them without altering the feature weights (b), versus using a partial matching normalized by only the minimum input cardinality (c). Each group above depicts two point sets containing one-dimensional features. (The top example contains one feature in each set, the remaining three examples contain two features in one set, and one in the other.) The same sets are matched for (a) and (b) and (c). A circle with a number in it represents a feature, and lines between features denote what is being matched in every group. The numbers on the lines denote the cost (a) or similarity (b,c) associated with that correspondence link. Total matching scores are displayed to the right for each set. Notice how outlier features skew the matching costs in (a), while (b) assigns the same similarity to all three bottom matches. In addition, the cost for (a) does not distinguish between the left set's feature distributions in the top two examples, while the similarity in (b) does reflect their differences. When normalizing by the minimum cardinality, there is absolutely no penalty for unmatched features (c); however, this matching does not obey the triangle inequality and thus does admit locality sensitive hash functions.

sponding distance for $1 - sim(x, y)$ is then

$$\begin{aligned} \mathcal{D}_p(\mathbf{X}, \mathbf{Y}; \pi) &= 1 - \mathcal{S}\left(\mathcal{M}(\mathbf{X}, \mathbf{Y}; \pi)\right) \\ &= 1 - \Big(\frac{1}{\min(|\mathbf{X}|, |\mathbf{Y}|)} \sum_{\mathbf{x}_i \in \mathbf{X}} \frac{1}{||\mathbf{x}_i - \mathbf{y}_{\pi_i}|| + 1}\Big). \end{aligned} \tag{5.12}$$

The partial match computed with this normalization—where there is no penalty for points not matched from the larger set—does not satisfy the triangle inequality, as we can see with a simple counter-example. For sets with one-dimensional features: $\mathbf{X} = \{[1]\}$, $\mathbf{Y} = \{[2]\}$, and $\mathbf{Z} = \{[1], [2]\}$, the pairwise distances are $\mathcal{D}_p(\mathbf{X}, \mathbf{Y}; \pi^*) = \frac{1}{2}$, $\mathcal{D}_p(\mathbf{X}, \mathbf{Z}; \pi^*) = 0$, and $\mathcal{D}_p(\mathbf{Z}, \mathbf{Y}; \pi^*) = 0$. This breaks the triangle inequality, since $\mathcal{D}_p(\mathbf{X}, \mathbf{Z}; \pi^*) + \mathcal{D}_p(\mathbf{Z}, \mathbf{Y}; \pi^*) < \mathcal{D}_p(\mathbf{X}, \mathbf{Y}; \pi^*)$. Therefore, a partial matching score normalized by the minimum input set size does not admit a locality sensitive hash function.

Chapter 6

Approximating the Optimal Matching in Practice

As described in Chapter 3, the pyramid match approximates the optimal correspondence-based matching between two feature sets. The theoretical bounds we can show are actually significantly weaker than what we observe in practice. In this chapter, I provide a series of experiments to empirically evaluate the approximation quality of the pyramid match with real and synthetic data sets. I evaluate the pyramid match using both uniform histogram bins and the vocabulary-guided bins, and I make a direct comparison between my partial matching approximation and the L_1 embedding of Indyk and Thaper [57]. Throughout these experiments, unless otherwise stated, I have employed the pyramid match using uniform bins.

6.1 Comparison with a Bijective Matching Embedding

I conducted experiments to evaluate how close the correspondences implicitly measured by the pyramid match are to the true optimal correspondences—the matching that results in the minimal summed cost between corresponding points. In order to work with realistic data but still have control over the sizes of the sets and the amount of clutter features, I established synthetic "category" models. Each model is comprised of some fixed number m' of parts, and each part has a Gaussian model that generates its d-dimensional appearance vector (in the spirit of the "constellation model" used by Fergus et al. [34] and others). Given these category models, I can then add clutter features, adjust noise parameters, and so on, simulating in a controlled manner the variations that occur with the sets of image patches extracted from an actual object. The appearance of the clutter features is determined by selecting a random vector from a uniform distribution on the same range of values as the model features.

I generated two data sets, one with equally-sized sets, and one with variable-sized sets. Every point set was drawn from one of two synthetic category models, with $m' = 35$ and $d = 2$. For the first data set, 50 point sets containing only the m' model features were

sampled from both of the two category models, for a total of 100 equally-sized point sets. For the other data set, the model feature sets were merged with a randomly generated number C of "extra" clutter features, for a total of 100 point sets with $m' + C$ features each, with C selected uniformly at random from $[0, 100]$.

I compared the pyramid match's outputs to those produced by the optimal partial matching obtained via a linear programming solution to the transportation problem [92], as well as those produced by the L_1 approximation of [57]. For both of the data sets, I computed the pairwise set-to-set distances using each of these three measures.

If an approximate measure is approximating the optimal matching well, we should find the ranking induced by that approximation to be highly correlated with the ranking produced by the optimal matching for the same data. In other words, the point sets should be sorted similarly by either method. We can display results in two ways to evaluate if this is true: 1) by plotting the actual costs computed by the optimal and approximate method, and 2) by plotting the rankings induced by the optimal and approximate method. Spearman's rank correlation coefficient R provides a good quantitative measure to evaluate the ranking consistency:

$$R = 1 - \frac{6 \sum_{i=1}^{N} (i - \hat{r}(i))^2}{N(N^2 - 1)}, \tag{6.1}$$

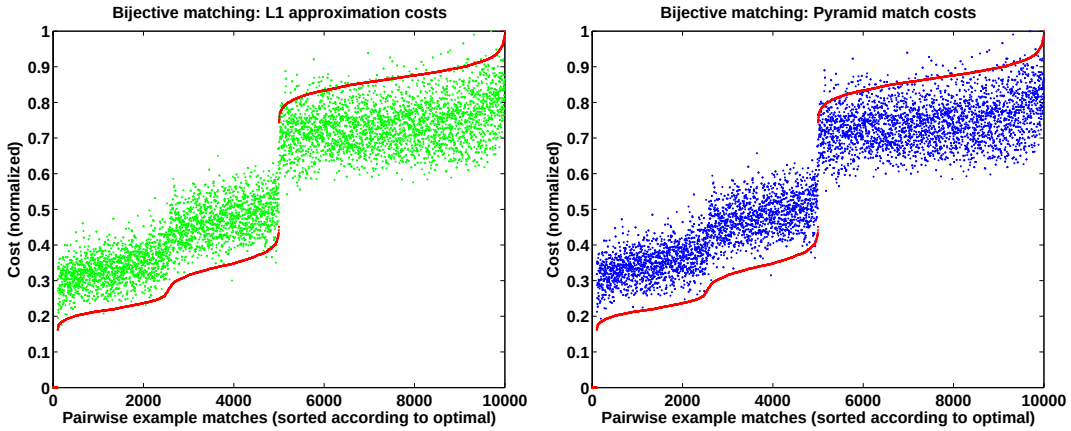
where i is the rank value in the true order and $\hat{r}(i)$ is the corresponding rank assigned in the approximate ordering, for each of the N corresponding ordinal values assigned by the two measures.

Figures 6-1 and 6-2 display both types of plots for the two data sets. Figure 6-1 displays plots corresponding to the data set with equally-sized sets, that is, for the bijective matching problem. Figure 6-2 displays plots corresponding to the data set with variable-sized sets, that is, for the partial matching problem.

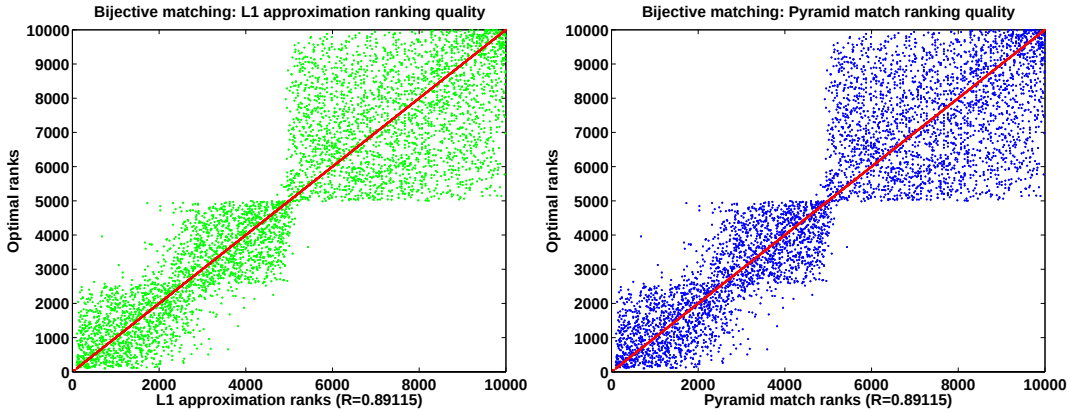
The top two plots in both figures show the normalized output costs from 10,000 pairwise set-to-set comparisons computed by the optimal matching (red in both), the pyramid match with $T = 3$ (blue, right), and the L_1 approximation, also with the number of random shifts $T = 3$ (green, left). Note that in these figures we plot cost (so the pyramid match weights are set to $w_i = d2^i$), and for display purposes the costs have been normalized by the maximum cost produced for each measure to produce values between $[0, 1]$. The cost values are sorted according to the optimal measure's magnitudes for visualization purposes. Note that the raw values produced by the approximate measures will always overestimate the cost; normalizing by the maximum cost value simply allows us to view them against the optimal measure on the same scale.

The bottom two plots in both figures display the rankings for each approximation plotted against the optimal rankings. The red diagonals in these plots denote the optimal performance, where the approximate rankings would be identical to the optimal ones. The R value displayed beneath the ranking quality plots refers to the Spearman rank correlation score for that experiment; higher Spearman correlations have points clustered more tightly along this diagonal.

Both approximations compute equivalent costs and rankings for the bijective matching

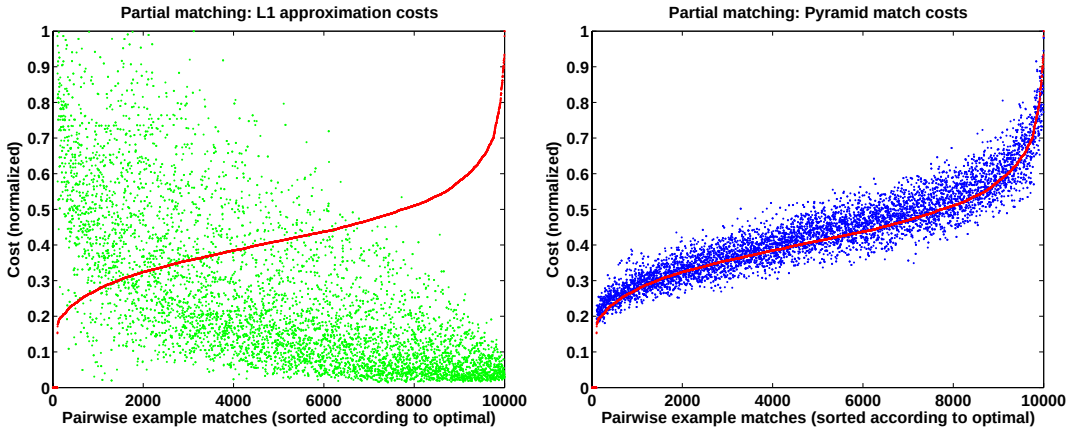


(a) Cost plots

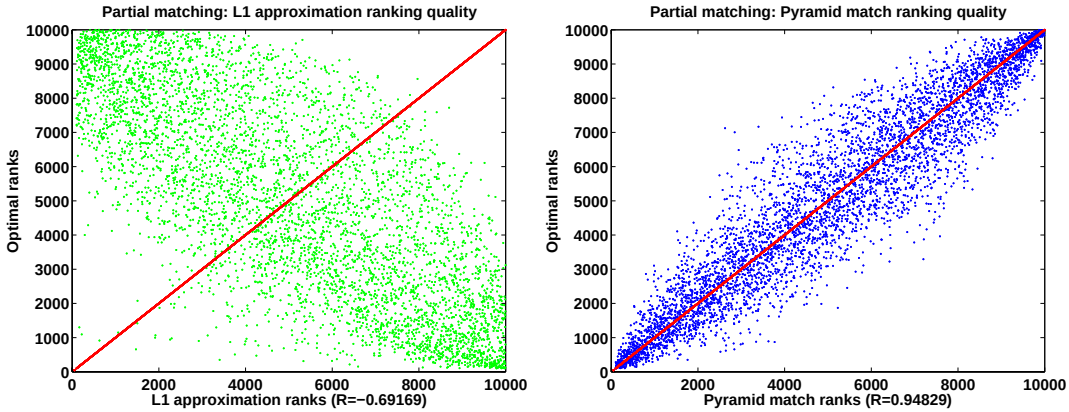


(b) Rank plots

Figure 6-1: Pyramid match and L_1 embedding comparison on bijective matchings (equally-sized sets). When all input sets are the same size, the pyramid match and the L_1 embedding give equivalent results. See text for details. (This figure is best viewed in color.)



(a) Cost plots



(b) Rank plots

Figure 6-2: Pyramid match and L_1 embedding comparison on partial matchings (variable-sized sets). The pyramid match approximates the optimal correspondences, even for input sets of unequal cardinalities. See text for details. (This figure is best viewed in color.)

case, as indicated by Figure 6-1. This is an expected outcome, since the L_1 distance over histograms is directly related to histogram intersection *if* those histograms have equal masses [104], as they do for the bijective matching test:

$$\mathcal{I}(H(\mathbf{Y}), H(\mathbf{Z})) = m - \frac{1}{2}||H(\mathbf{Y}) - H(\mathbf{Z})||_{L_1}, \quad \text{if } m = |\mathbf{Y}| = |\mathbf{Z}|. \tag{6.2}$$

The structure along the diagonals in part (b) of Figure 6-1 reflects the fact that two types (categories) of point sets were present in the data, causing items of the same category to block together when they are sorted by matching cost. The square-shaped cluster in the upper right portions of the plots show that while the approximate measures do not distinguish between all examples within a category precisely the way the optimal measure does, they do both consider all items within one category to be most distant from an example drawn from the other category. Similarly, there are discontinuities in part (a) due to distance clusters caused by drawing point sets from two distinct category models.

However, for the partial matching case (Figure 6-2), the L_1 approximation fails because it can handle only sets with equal cardinalities and requires all points to match to something. On the other hand, the pyramid match can also approximate the partial matching for the unequal cardinality case: its matchings continue to follow the optimal matching's trend since it does not penalize outliers, as evidenced by the tightly clustered points along the diagonal in the bottom right ranking quality plot. Please note that this experiment is something of a "sanity check", since we would expect the method of Indyk and Thaper to fail for unequally-sized sets; it allows only equal-mass point sets as input, and is not meant to compute partial matchings. I have performed this same experiment using data generated from a uniform random point model, and the outcome was similar.

6.2 Impact of Clutter on the Approximation Accuracy

In this section I evaluate the accuracy of the partial matching computed by the pyramid match with uniform bins in the presence of varying amounts of clutter.

In order to control the amount of clutter in the input point sets, I again establish realistic synthetic object category models. Each category model is defined by a certain number of parts, and each part has a Gaussian model that generates its appearance descriptor vector. A number of example point sets are sampled from the category models using the Gaussian parameters. Then, a controlled number of features representing clutter can be added to each point set. The appearance of a clutter feature is determined by selecting a random vector from a uniform distribution on the same range of values as the model features.

By adding increasing amounts of clutter to the point sets sampled from the model distributions, I can evaluate both how well a partial matching is able to resist clutter (i.e., how well it can recognize similarity between point sets from the same model in spite of extraneous clutter features), in addition to how well the pyramid match approximates the tolerance of the optimal matching in this regard. There are a few metrics to assess this: distributions of the raw error between the costs estimated by the pyramid match and the optimal matching, their Spearman ranking correlations, and the *relevant ranking* quality of the optimal and

approximate measures. The Spearman ranking coefficient was used in the previous experiments; it reflects how similarly two measures sort the collection of point sets. The raw cost error between an approximate matching $\hat{\pi}$ and the optimal matching π^* is defined as the absolute difference between their associated costs:

$$\begin{aligned} E_{raw}\left(\mathcal{M}(\mathbf{X},\mathbf{Y};\hat{\pi}),\mathcal{M}(\mathbf{X},\mathbf{Y};\pi^*)\right) &= \left|\mathcal{C}(\mathcal{M}(\mathbf{X},\mathbf{Y};\hat{\pi})) - \mathcal{C}(\mathcal{M}(\mathbf{X},\mathbf{Y};\pi^*))\right| \\ &= \left|\sum_{\mathbf{x}_i\in\mathbf{X}} ||\mathbf{x}_i - \mathbf{y}_{\hat{\pi}_i}|| - \sum_{\mathbf{x}_i\in\mathbf{X}} ||\mathbf{x}_i - \mathbf{y}_{\pi^*_i}||\right|. \end{aligned} \quad (6.3)$$

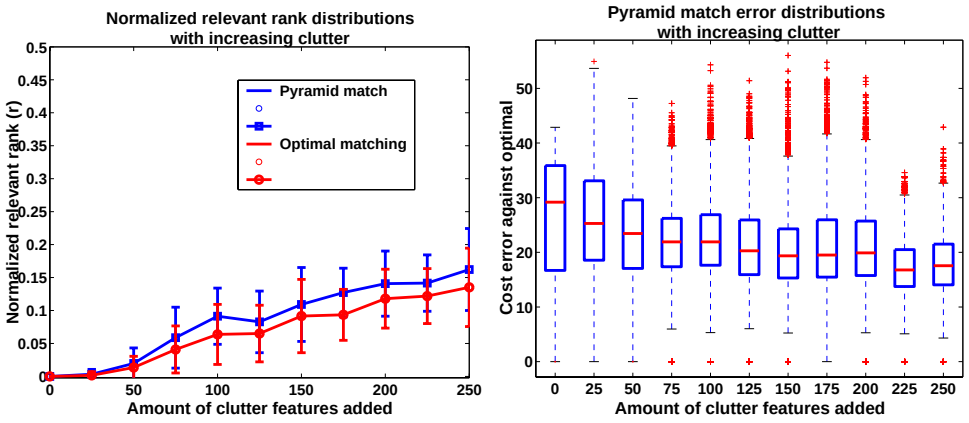
Notice that both the Spearman rank correlation and the absolute cost error do not take into account any labels associated with the input sets. They solely evaluate how closely the approximate measure estimates the optimal matching costs, or how similarly it ranks a collection of inputs. For this experiment I also exploit the labels attached to each input set to evaluate the actual *quality* of similarity measured by a matching, whether it is approximate or optimal. That is, we can evaluate how well the partial matching serves to group point sets that were drawn from the same model distribution; the partial matching is best if it finds the highest similarity among sets representing instances of the same object category, and vice versa. To quantify this, I use the *normalized average rank* $\bar{r}$:

$$\bar{r} = \frac{1}{NN_R}\left(\sum_{i=1}^{N_R} r_i - \frac{N_R(N_R-1)}{2}\right), \quad (6.4)$$

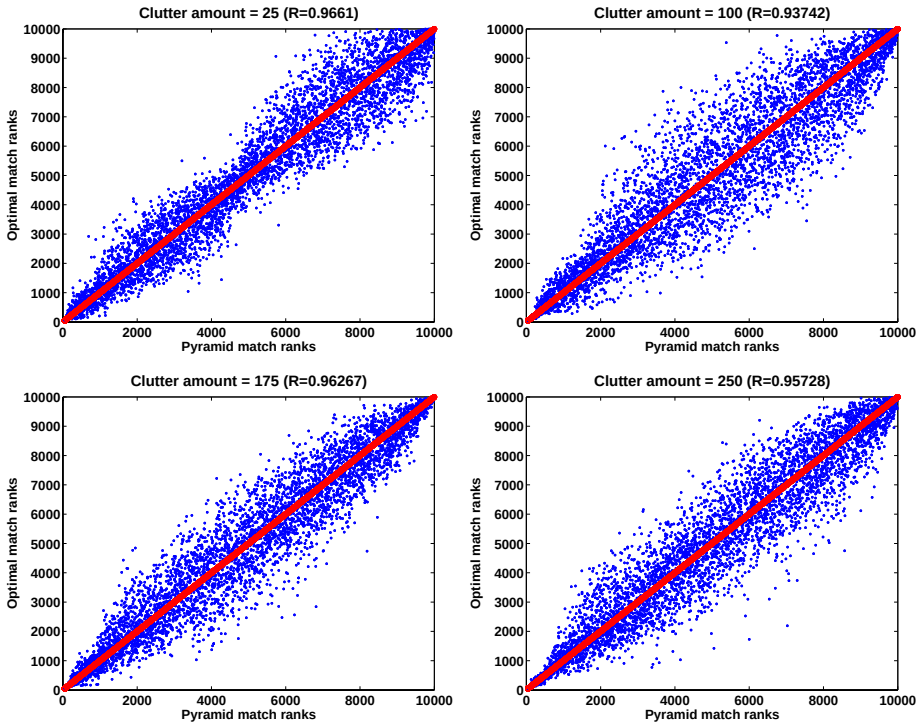
where r_i is the rank assigned to the ith relevant example (set), N_R is the number of relevant examples for a given example, and N is the total number of examples. The normalized rank is 0 for perfect performance (i.e., when all relevant examples in the data set are ranked as an input's nearest neighbors), and it approaches 1 as performance worsens; a random ranking results in an expected normalized rank of 0.5 [82]. In this case, relevant examples are those sets that were sampled from the same category model.

I generated 50 examples for two synthetic category models, each of which was defined by a set of $m = 50$ features with $d = 2$, for a total of 100 point sets. Then I added increasing amounts of clutter features to every set and computed the resulting pairwise matching scores for all sets according the pyramid match and the optimal partial matching. For a given run, the amount of clutter added to each set varied uniformly between 0 and M; values of M between 0 and 250 were tested, in increments of 25.

Figure 6-3 displays the results. The left plot in part (a) shows the distribution of the normalized relevant ranking scores assigned by the optimal matching (red) and the pyramid match (blue) across the different values of M. The points reflect mean values, and the bars are standard deviations for all 100×100 matching scores. This plot reflects how successfully the partial matching recognizes the similarity between relevant examples, even when significant amounts of clutter must be ignored to do so. Both the approximate and optimal measure have perfect rankings ($\bar{r} = 0$) with $M = 25$, which is the case where the clutter comprised as many as half the number of true model features. Even when the amount of clutter equals the amount of model features ($M = 50$), the relevant ranking is very strong (mean $\bar{r} < 0.025$).



(a) Pyramid match error as amount of clutter increases



(b) Pyramid match ranking quality as clutter increases

Figure 6-3: Evaluating the impact of clutter features on the pyramid match's approximation accuracy. See text for details.

Beyond that, the relevant rankings deteriorate slowly for both measures as more clutter is added. This is because once the clutter features in each set outnumber the model features, it may be possible for a better partial match to be formed entirely between clutter features from two sets, causing the relevant model features to be ignored. By the definition of a partial matching, this is actually correct behavior; however, as the plot shows, it is rare rather than typical for this degenerate case to occur since the distribution of the clutter features is wider than that of the model features. Additionally, the rankings are always much better than random ($\bar{r} = 0.5$, the maximum value on the vertical axis), with the optimal measure only slightly outperforming the pyramid match for values of $M \geq 75$.

The righthand plot of Figure 6-3 (a) shows the raw error distributions for each value of M, displayed as boxplots. The lines in the center of the boxes denote the median value, the top and bottom of each box denotes the upper and lower quartile values; the dashed lines show the extent of the rest of the errors. This shows that the error of the pyramid match versus the optimal matching remains fairly steady, regardless of the amount of clutter. This is reasonable, since nothing about the approximation factor relates to the number of features in a set.

Part (b) of the same figure shows four ranking plots covering the spectrum of amounts of clutter tested, with the associated Spearman correlations (R) printed in the title of each. Just as the pyramid match's absolute error is consistent across amounts of clutter, the similarity of its ranking to the optimal matching's ranking remains steady.

In Chapter 5 I showed how to perform sub-linear time hashing with the pyramid match. With point sets that are embedded as histogram pyramids, random hyperplane hashing provides bounded approximate similarity search. In this case, similarity is measured by the partial matching between point sets, normalized according to the product of the sizes of both input sets. As discussed in Section 5.2.2, while this normalization scheme does admit a locality-sensitive pyramid match hashing function, normalization by the minimum input cardinality would not.

With the product normalization, the *number* of unmatchable features will have some effect on the matching scores. However, the degree to which the unmatchable (or "outlier") features *differ* from the matched features will not affect the matching scores, meaning that pyramid match hashing is robust to increasingly distant outlier features. In contrast, bijective matchings computed over point sets that have been re-weighted to achieve total equal masses are not robust to increasingly distant outliers. (Recall Figure 5-2 in Chapter 5.)

These properties hold by definition, and I verify them empirically here with an experiment with data similar to that used above. I again generated 50 examples for two synthetic category models, each of which was defined by a set of $m = 35$ features with $d = 2$, for a total of 100 point sets. I computed pairwise similarities using the pyramid match normalized by the product of the input sets' cardinalities, pairwise similarities using the optimal matching and the same normalization, and pairwise distances based on the bijective matching approximation of [57]. To apply the bijective matching to unequally-sized sets, points in a set were weighted so that all weights summed to one (as outlined in the pseudo-code in Procedure 1 in Chapter 5).

Then I added to every set up to 100 clutter features having a value range bounded by

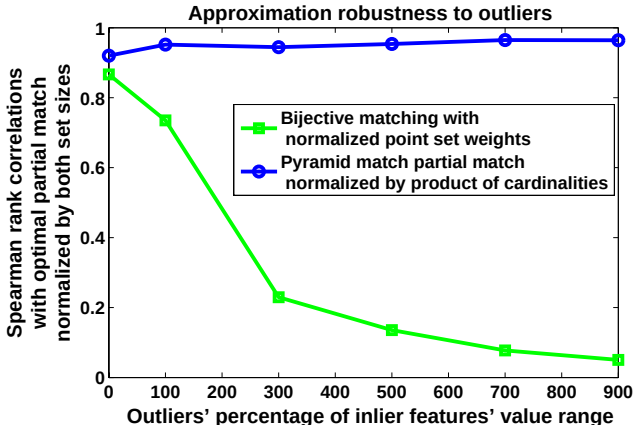


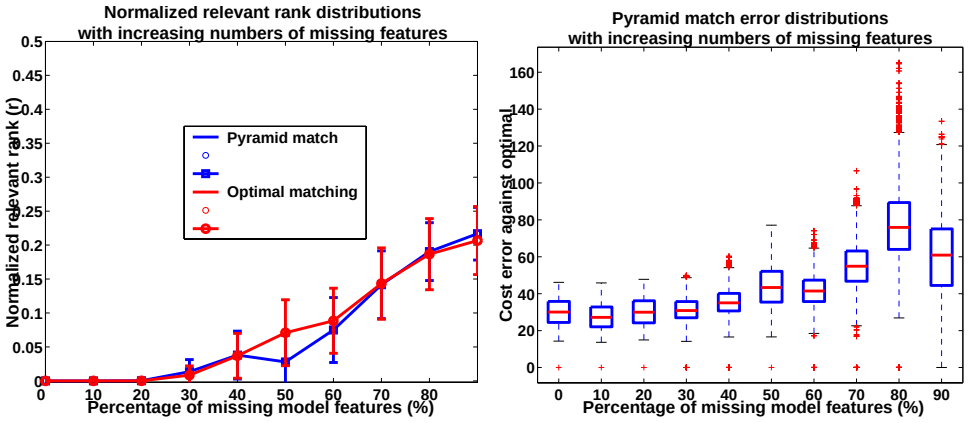
Figure 6-4: A partial match normalized by the product of the sizes of both input sets will remain robust to distant outlier clutter features (blue circles). This property cannot be simulated by re-weighting point sets and computing a bijective matching (green squares).

a percentage Q of the inlier features' value range, and re-computed the resulting pairwise matching scores. I tested for values of Q ranging from 100% to 1000%, in increments of 200. (When $Q = 100\%$, the inlier and outlier features have the same value range.) Figure 6-4 shows the results, with ranking quality quantified by the Spearman correlation coefficient. The two left-most points on the plot correspond to matchings with equally-sized sets and no clutter. The remaining points correspond to matchings with increasingly more distant clutter or outlier features. The pyramid match scores normalized by the sizes of both input sets remain robust to the addition of stronger outlier features (blue circles), whereas the bijective matching must incorporate the distance of the outlier features in its matching and suffers as that distance increases (green squares). For no outlier features, the results are similar.

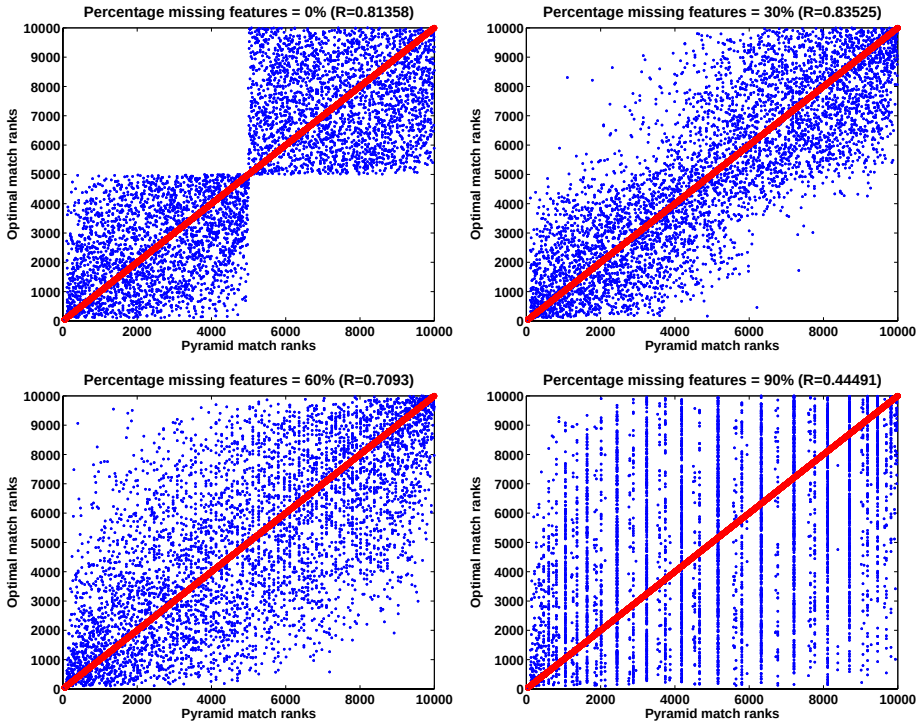
6.3 Impact of Missing Features on the Approximation Accuracy

In practice, we can expect some of an object's local features to be missing in different image descriptions due to viewpoint changes, occlusions from other objects, or simply unreliable feature detections. Therefore, it is important to consider the ability of a partial match to handle inputs that may contain only a portion of the features belonging to a particular category's model. In this section I present an experiment designed to evaluate the effect of missing features on the pyramid match approximation accuracy and the ranking quality of a partial match in general.

As in the previous section, I established two synthetic category models containing $m = 50$ parts each, with features of dimension $d = 2$. I again sampled 50 sets of points from each model. Then, I computed the pairwise matchings for all the sets, but first altered each set by removing some percentage P of its model features. I tested for values of P between 0% and 90%, at increments of 10.



(a) Pyramid match error as number of missing features increases



(b) Pyramid match ranking quality as number of missing features increases

Figure 6-5: Evaluating the impact of missing features on the pyramid match's approximation accuracy. See text for details.

Figure 6-5 displays the results. The format of this figure is identical to the format of Figure 6-3, except that now the parameter we are testing (the horizontal axis in both plots in part (a)) is the percentage of missing features per set. The left plot in part (a) shows that even when as many as 30% of an object's features are missing, the impact on the relevant ranking quality is negligible ($\bar{r} < 0.025$). Then as we remove increasing portions of the model's description, it becomes increasingly difficult for the matching to discriminate between the two categories, as seen by the increasing relevant rank scores. This is reasonable, since the pool of appearance features for both categories does overlap, and so only observing a portion of the features can make the underlying model ambiguous. Again, however, the pyramid match is very consistent with the optimal partial matching in terms of producing relevant rankings.

The boxplots in the righthand plot in part (a) of the same figure show that the pyramid match error distribution remains compact and low for percentages of missing features up to about $P = 70\%$. When 80% or 90% of the features are missing, the pyramid match's cost estimates suffer a bit more, though the relevant rankings in the lefthand plot suggest that the increase in raw error does not impact the quality of the relative ranks.

Finally, the four plots in part (b) of the same figure illustrate the approximate matching's ranking similarity to the optimal matching across the spectrum of amounts of missing features tested, with the associated Spearman correlations (R) printed in the title of each. The top left plot in this group shows the rankings when there are no features missing. Here, the Spearman correlation is only $R = 0.81$, meaning the rankings are fairly highly correlated, but that the pyramid match does not exactly order items *within* a given category as the optimal measure would, as reflected by the two blocks along the diagonal. However, in terms of relevancy, the ranking is perfect. In other words, all items of the same category are ranked by the pyramid match as more similar than all items of a different category, regardless of how the rankings within the category may be shuffled versus the optimal output.

The remaining three plots in this group show how the precise ranking quality is still strong for $P = 30\%$ and $P = 60\%$ (R values of 0.84 and 0.71), but then significantly declines when $P = 90\%$ of the features are missing per set. The vertical line structures in this plot reveal that the pyramid match has assigned tied ranks for a large number of the sets, indicating that when only 10% of the model features are present, they can match equally well to a number of other sets drawn from either category.

6.4 Impact of the Feature Dimension on the Approximation Accuracy

In the remaining experiments in this chapter, I provide results to empirically demonstrate my method's accuracy and efficiency on real image data, and I compare the accuracy resulting from uniform bin pyramids to that obtained with vocabulary-guided pyramids. As in the previous experiments, I directly evaluate the matching scores and correspondence fields that are implied by the pyramid match against those produced with the optimal least-cost matching.

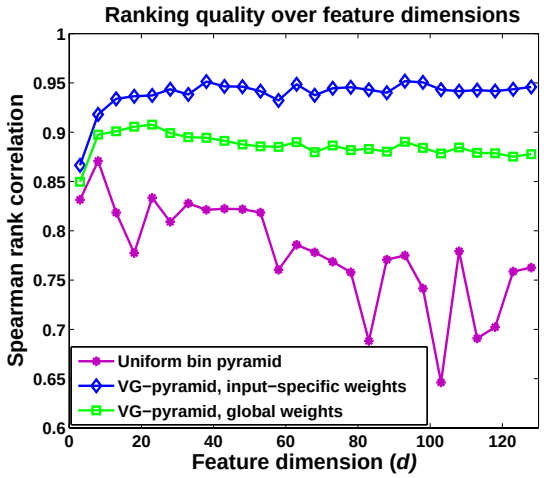
In this section in particular I evaluate the accuracy of the approximate matching as the dimension d of the feature space F increases.

For the following experiments, I extracted local SIFT [73] features from images in the ETH-80 database [69], producing an unordered set of about $m = 256$ vectors for every example. In this case, F is the space of SIFT image features. I used randomly sampled features from 300 of the images to build the vocabulary-guided pyramid, and 100 images were used to test the matching. In order to test across varying feature dimensions, I also used some training features to establish a PCA subspace that was used to project features onto varying numbers of bases. For each feature dimension, I build a vocabulary-guided pyramid with $k = 10$ and $L = 5$, encode the 100 point sets as pyramids, and then compute the pairwise matching scores with the uniform-binned pyramid match, the vocabulary-guided pyramid match, and the optimal least-cost matching. For the vocabulary-guided pyramid match I evaluate both weighting schemes: the weights based on the "global" diameters of the histogram bins (Eqn. 4.3 (a)) and the input-specific weights that exploit the features' distances to the bin centers (Eqn. 4.3 (b)).

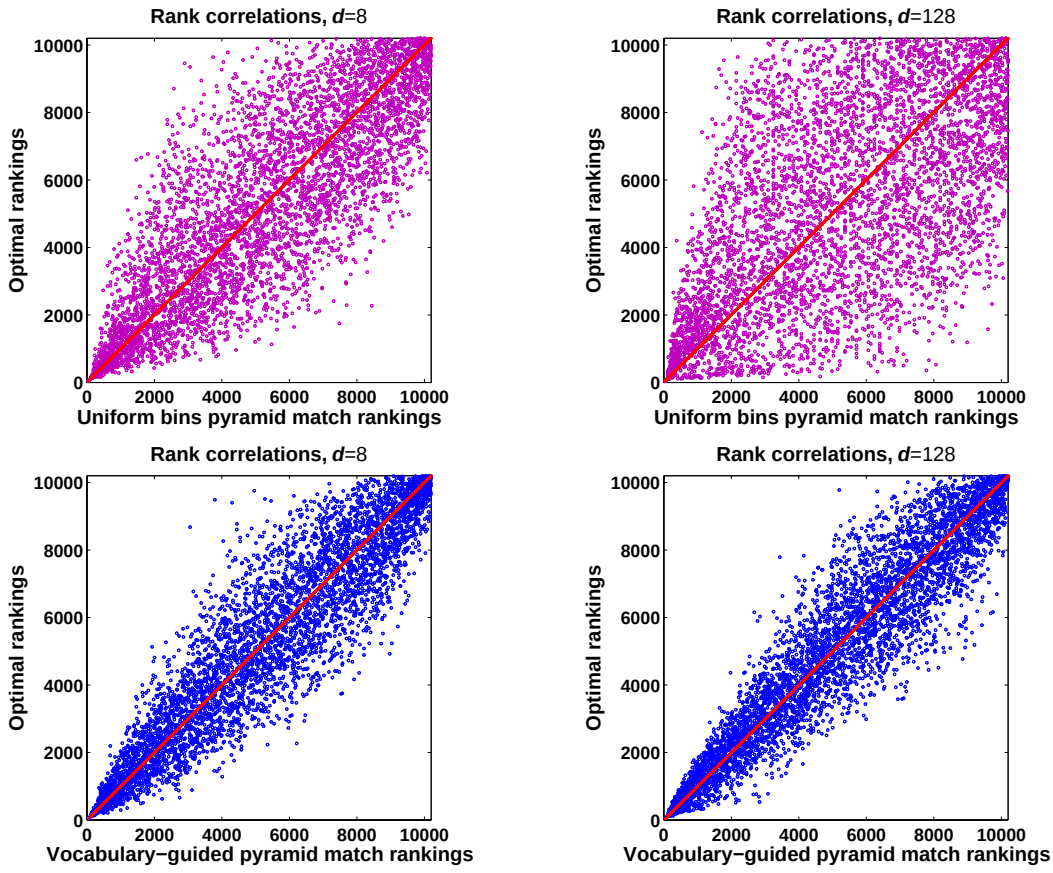
The top plot in Figure 6-6 (a) shows the Spearman correlation scores against the optimal measure for the vocabulary-guided (VG) and uniform bin pyramid matchings. The results from varying feature dimensions are shown for the 10,000 pairwise matching scores for the 100 test sets. The ranking quality of the input-specific weighting scheme (blue points) is somewhat stronger than that of the "global" bin diameter weighting scheme (green points); by definition, the input-specific weights can estimate a tighter limit on the distance between two points in a bin, yielding a more precise sum of edge costs. Nonetheless, the global weighting scheme does produce consistently accurate results across feature dimensions—the average correlation with the optimal match is 0.89—and as discussed in Chapter 4, this choice of weights does have a small computational advantage over the input-specific weights, and admits a Mercer kernel.

While the vocabulary-guided pyramid match remains consistently accurate (correlations above 0.94 with input-specific weights) for high feature dimensions, the accuracy of the approximation with uniform bins does degrade almost linearly as expected. The four plots in part (b) of Figure 6-6 display the actual ranks computed for both approximations (VG with input-specific weights and the uniform-binned pyramid match) for two of the 26 dimensions summarized in the plot in part (a). For low-dimensional features, both uniform and vocabulary-guided pyramids perform fairly comparably; however, for the full 128-D features, the vocabulary-guided pyramid match is far superior (right column of part (b)). The optimal measure requires about $1.25s$ per match, while the pyramid match approximation is about $2500x$ faster at $5 \times 10^{-4}s$ per match.

The well-known "curse of dimensionality" gives insights as to why the uniform bin pyramid in practice suffers noticeably for high dimensions. For a pyramid matching to work well, the gradation in bin sizes up the pyramid must be such that at most levels of the pyramid we can capture distinct groups of points to match within the bins. That is, unless all the points in two sets are equidistant, the bin placement must allow us to match very near points at the finest resolutions, and gradually add matches that are more distant at coarser resolutions. In low dimensions, both uniform or data-dependent bins can achieve this. In high dimensions, however, uniform bin placement and exponentially increasing bin



(a) Ranking quality across dimensions



(b) Example rankings

Figure 6-6: Comparison of optimal and pyramid match approximate matching rankings on real image data. *Part (a):* The set rankings produced with the vocabulary-guided (VG) pyramid match (blue diamonds and green squares) are consistently accurate for increasing feature dimensions, while the accuracy when using uniform bins (purple circles) degrades with the feature dimension. *Part (b):* Example rankings for both approximations at $d = 8$ (left column) and $d = 128$ (right column).

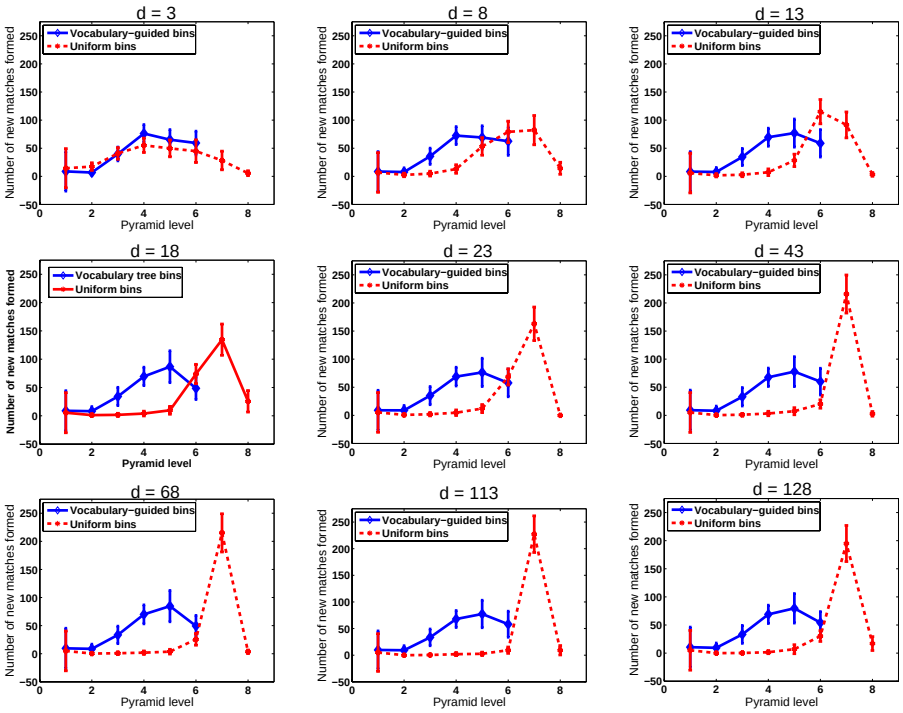


Figure 6-7: Number of *new* matches formed at each pyramid level for either uniform (dashed red) or vocabulary-guided (solid blue) bins for sets with increasing feature dimensions. Points represent mean counts at each level over 10,000 matches; bars denote standard deviations. At low dimensions, both partition styles gradually collect matches up the pyramid. However, in higher dimensions with the uniform partitioning, points begin sharing a bin "all at once", i.e., at the same pyramid level. In contrast, the vocabulary-guided bins accrue new matches consistently spread across levels—even for high dimensions—since the decomposition is tailored to where points cluster in the feature space.

diameters fail to capture such a gradation: once any features from different point sets are close enough to match (share bins), the bins are large enough that almost *all* of them match. The matching score is then approximately the number of points weighted by a single bin size. In contrast, because the vocabulary-guided pyramid tailors the feature space partitions to the distribution of the data, even in high dimensions we will have a gradual increase in bin size across levels, and in effect will obtain more discriminating implicit matches.

Figure 6-7 confirms this intuition, again using the ETH-80 image data from above. Note that there are different numbers of total levels for the two types of pyramids in these plots (L=6 and L=8); while the number of levels is set automatically based on the feature space aspect ratio D for the uniform bin pyramids, for the vocabulary-guided pyramid the number of levels is a parameter that dictates how fine grained the hierarchical clustering should get.

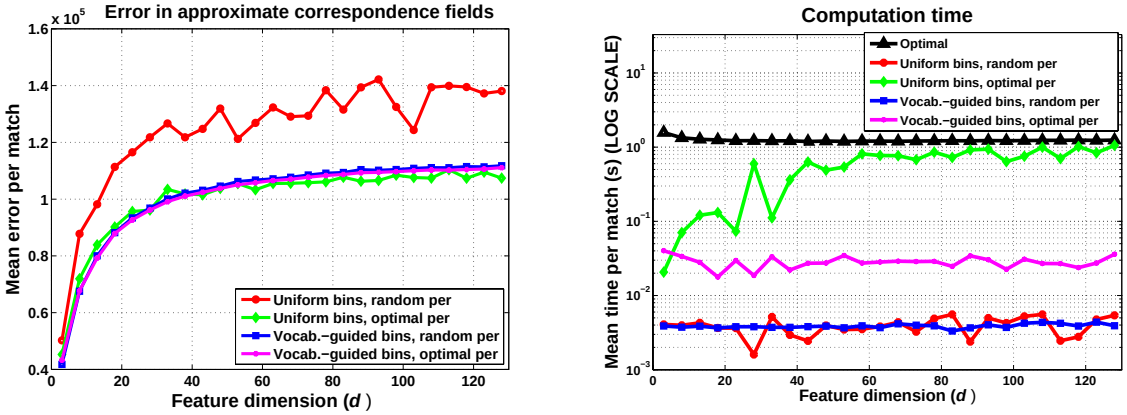


Figure 6-8: Comparison of correspondence field errors (left) and associated computation times (right) when using uniform or vocabulary-guided pyramid bins. This figure is best viewed in color. (Note that errors level out with d for all methods due to PCA; the most variance is in the first dimensions for all vectors.)

6.5 Approximate Correspondence Fields

For many applications, the scalar matching score alone is desired and the implicit matching is sufficient. However, we may have use for the explicit correspondence field that the matching reflects. For example, when matching sets of image features, knowing which component features were mapped to which could allow us to localize an object in a scene.

For the same image data, I used the pyramids to form an explicit matching as described in Section 3.2.3, and compared the correspondences between features induced by my matching to those produced by the globally optimal measure. For comparison, I computed correspondences with both uniform-bin pyramids and vocabulary-guided pyramids. Using the notation of Chapter 3, I measure the error of the correspondences of an approximate matching $\hat{\pi}$ as the sum of the errors at every link in the field:

$$E_{corr}\left(\mathcal{M}\left(\mathbf{X},\mathbf{Y};\hat{\pi}\right),\mathcal{M}\left(\mathbf{X},\mathbf{Y};\pi^{*}\right)\right)=\sum_{\mathbf{x}_i\in\mathbf{X}}||\mathbf{y}_{\hat{\pi}_i}-\mathbf{y}_{\pi_i^{*}}||. \tag{6.5}$$

Figure 6-8 compares the correspondence field error and computation times for the vocabulary-guided and uniform pyramids. For each approximation, there are two variations tested: in one, an optimal assignment is computed for all points in the same bin; for the other, a random assignment is made. The left plot shows the mean error per match for each method, and the right plot shows the corresponding mean time required to compute those matches.

The computation times are as we would expect: the optimal matching is orders of magnitude more expensive than the approximations (black line with triangles on right plot, which plots time on a log scale). Using the random assignment variation, both approximations have negligible costs, since they simply choose any combination of points within a bin.

It is important to note that in high dimensions, the time required by the uniform bin pyramid with the optimal per-bin matching approaches the time required by the optimal matching

itself. This occurs for similar reasons as the poorer matching score accuracy exhibited by the uniform bins, both in the left plot and above in Figure 6-6; since most or all of the points begin to match at a certain level, the pyramid does not help to divide-and-conquer the computation, and for high dimensions, the optimal matching in its entirety must be computed. In contrast, the expense of the vocabulary-guided pyramid matching remains steady and low, even for high dimensions, since the data-dependent pyramids better divide the matching labor into the natural segments in the feature space.

For similar reasons, the errors are comparable for the optimal per-bin variation with either the vocabulary-guided or uniform bins. The vocabulary-guided bins divide the computation so it can be done inexpensively, while the uniform bins divide the computation poorly and must compute it expensively, but about as accurately. Likewise, the error for the uniform bins when using a per-bin random assignment is very high for any but the lowest dimensions (red line on left plot), since such a large number of points are being randomly assigned to one another. In contrast, the vocabulary-guided pyramid bins actually result in similar errors whether the points in a bin are matched optimally or randomly (blue and pink lines on left plot). This again indicates that by tuning the bins to the data's distribution, the vocabulary-guided pyramid achieves a more suitable breakdown of the computation, even in high dimensions.

6.6 Summary

In this chapter I have shown the ability of the pyramid match to compute partial matchings on real and synthetic data that approximate the optimal partial match at a significantly lower computational cost. I have demonstrated the significant conceptual and practical difference between the pyramid match and a related approximation developed by Indyk and Thaper [57]. In a series of experiments I systematically assessed the pyramid match's approximation accuracy under three important conditions: increasing feature dimension, clutter, and missing features.

I demonstrated that a data-dependent hierarchical decomposition of the feature space can result in more accurate correspondence fields than using a uniform decomposition for high-dimensional features. By taking advantage of the underlying structure in the feature space, the pyramid match can remain consistently accurate and efficient for high-dimensional inputs on real image data. The experiments also showed that even with significant amounts of clutter or missing features, the pyramid match closely approximates the optimal partial matching, in terms of both actual cost estimates and relevant rankings.

Chapter 7

Content-Based Image Retrieval

In this Chapter I provide content-based image retrieval results using approximate correspondences between sets of local features, together with sub-linear time approximate similarity search using Locality-Sensitive Hashing. With the matching approximations it is feasible to quickly retrieve similar shapes and images from large databases—an ability which has applications in various example-based vision systems—and the constraint on input feature set size from which other techniques suffer is eliminated.

I experiment with several large data sets and compare against existing techniques, including the Earth Mover's Distance (EMD). In Section 7.1 I present results for contour or shape matching, while in Section 7.2 I focus on appearance-based local descriptions. In this chapter I only consider matches between sets of equal cardinality, and I use the L_1 embedding developed in [57], not the partial match described in Chapter 3. I initially reported the results in this chapter in [41, 43] and [42], which pre-dated my development of the partial match described in Chapter 3. Alongside the correspondence-based set matching approach, on a series of data sets I also evaluate the two main alternative image matching approaches reviewed in Chapter 2: voting and a bag-of-prototypes method.

7.1 Contour Matching with Local Shape Features

In this section, I first describe each of the data sets and feature representations used in the shape-based retrieval experiments. Then I report on the matching performance of my approach, followed by a discussion regarding its efficiency.[1]

7.1.1 Data Sets and Representation

I have run contour matching experiments on databases of human figures and handwritten digits. The human figure database contains 136,500 images of synthetic human figure contours in random poses that were generated with a computer graphics package called

[1] I first reported the results in this section in [41] and [43].

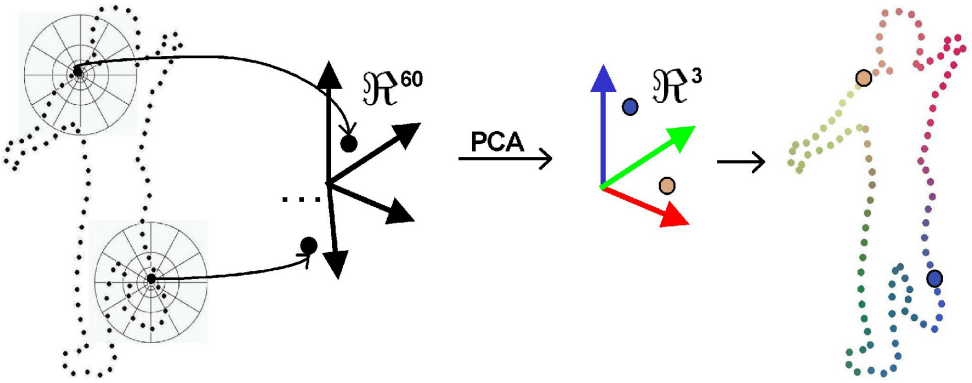


Figure 7-1: Projecting histograms of contour points onto the shape context subspace. The points on the human figure on the right are colored according to their 3-D shape context subspace feature values.

Poser [21]. I query this database with a separate test set of 7000 synthetic human figure images, and a test set of 1000 real images from a single human subject in various poses. The digit database we use is a publicly available benchmark data set, the MNIST database [80], which has been used to test many different pattern recognition algorithms. It contains 60,000 training images and 10,000 test images of digits written by about 250 different people. For the human figure images, we uniformly sampled 200 points from each silhouette contour, and for the digits we sampled 100 edge points per example. Since in these experiments we can sample equal numbers of contour points from every shape, we match the equally-sized sets with the L_1 approximation described by Indyk and Thaper in [57].

Shape Context Subspace Features

To represent shapes in terms of their local parts, I employ the local *shape context* descriptor of Belongie et al. [7]. The shape context feature at a single contour point is a log-polar histogram of the coordinates of the rest of the point set, measured using the reference point as the origin.

While matching with the full shape context feature is possible with the approximate matching, a low-dimensional feature descriptor is desirable in these experiments, since we are using a uniform bin approximation and can expect the matching accuracy to decline with the feature dimension d.[2] Thus we find a low-dimensional feature subspace based on a large sample of the shape context histograms, and then compute the pyramids in the domain of this subspace. The subspace is constructed from features sampled from the database of contours to which the method will be applied. All contour features (from the database items and novel queries alike) are then projected onto the subspace, and the approximate matching is performed in the domain of a small number of their subspace coordinates (see Figure 7-1). We use principal components analysis (PCA) to determine the set of bases that define this "shape context subspace".

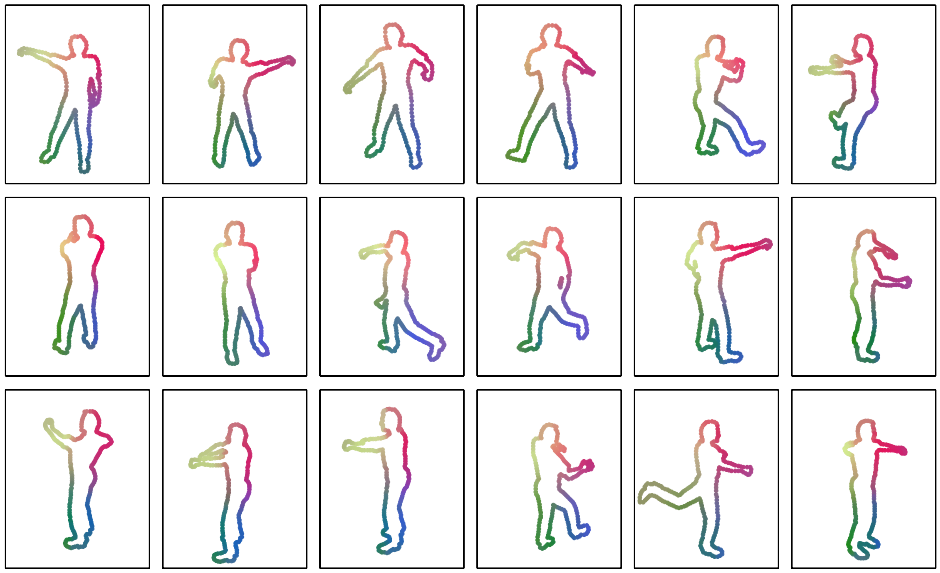
[2]The experiments in this chapter pre-dated my development of the vocabulary-guided pyramid match (described in Chapter 4); it would provide another way to address this issue.

I found that a very low-dimensional subspace was able to capture much of the local contour variation in these data sets. Figure 7-2 gives examples of the shape context subspace for human figures and handwritten digits. Figure 7-3 measures its expressiveness as a function of feature dimension, as compared to a higher-dimensional raw point histogram.

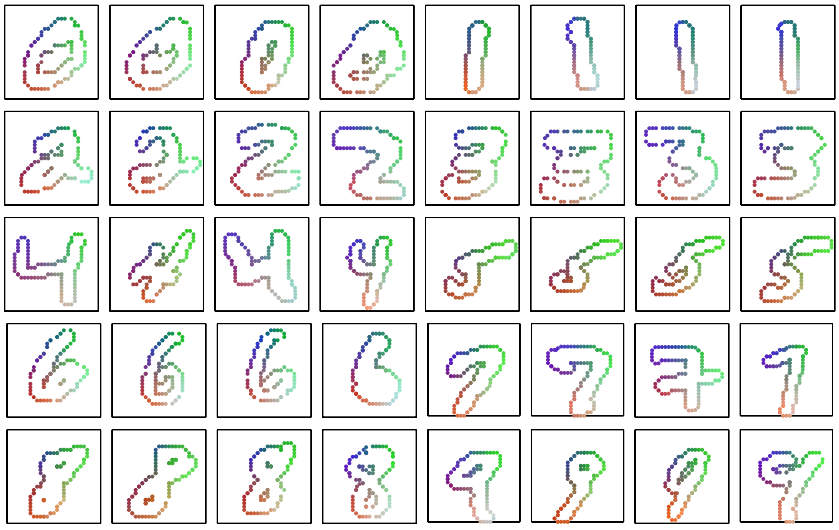
Each contour is described with an unordered set of shape context subspace features. I construct shape context subspaces from 5×12 log-polar histograms extracted from the training sets, using samples of 855,600 and 816,665 histograms from the human figure and handwritten digit data sets, respectively. The representation of a novel contour is determined by projecting its shape context histograms onto the low-dimensional subspace. This representation is translation invariant, and the scale of the shape context histograms initially extracted from the data is determined from the mean inter-point distance per shape.

There are several tradeoffs that must be considered when selecting d, the number of subspace dimensions to use for the shape context subspace features. The larger d is, the more exactly the projection coordinates will capture the original histogram feature extracted from the image (i.e., the smaller the PCA reconstruction error will be). However, larger d values will increase both the distortion that is induced by the embedding, as well as the complexity of computing the embedding. The dimension of the embedded vectors—and the time required to compute the L_1 distances—are also directly affected by d. Moreover, higher-dimensional subspace projections, though strictly capturing the original data more faithfully, will not necessarily provide the most meaningful representation of the feature; one purpose of reducing the features' dimensionality with PCA is to distill the most significant modes of variation within the high-dimensional feature and use only those most discriminating dimensions as the descriptor. In fact, since we have a quantitative task by which retrievals may be judged (k-nearest-neighbor classification performance), it is more appropriate to choose d with this objective than to select it solely on the number of eigenvalues with the highest magnitude (a standard approach).

Ultimately, the optimal setting of d—the setting that preserves both descriptiveness and efficiency—will depend on the data set. Thus in my experiments I determine the best setting by looking directly at the relationships between d and both recognition performance and retrieval efficiency. Using a reserved portion of the training set to query the remaining training examples, I record the correct k-NN classification rate, embedding times, and query times achieved with each increasing value of d. While seeking a good setting for d in this way, I measure relative query times in terms of an exhaustive L_1 search on embedded vectors (i.e., without LSH). This ensures both that the best possible classification performance is observed, and that the query times are fairly uniform across examples for the same d, so as not to be influenced by outlier cases that may occur with LSH. Plotting the recognition rates, embedding times, and query times then reveals the tradeoffs between complexity and classification performance and allows us to choose an appropriate value of d to be applied for the actual test examples (see Figure 7-4). Using this process, I found that for the human figure and MNIST digit data sets, a 2-D projection adequately captured the descriptive power of the shape context feature and produced good contour matches, yet also yielded very efficient embedding and query complexities.



(a) Human figure database



(b) Handwritten digits database

Figure 7-2: Visualization of feature subspace constructed from shape context histograms for two different data sets. The RGB channels of each point on the contours are colored according to its histogram's 3-D PCA coefficient values. Set matching in this feature space means that contour points of similar color have a low matching cost, while highly contrasting colors incur a high matching cost.

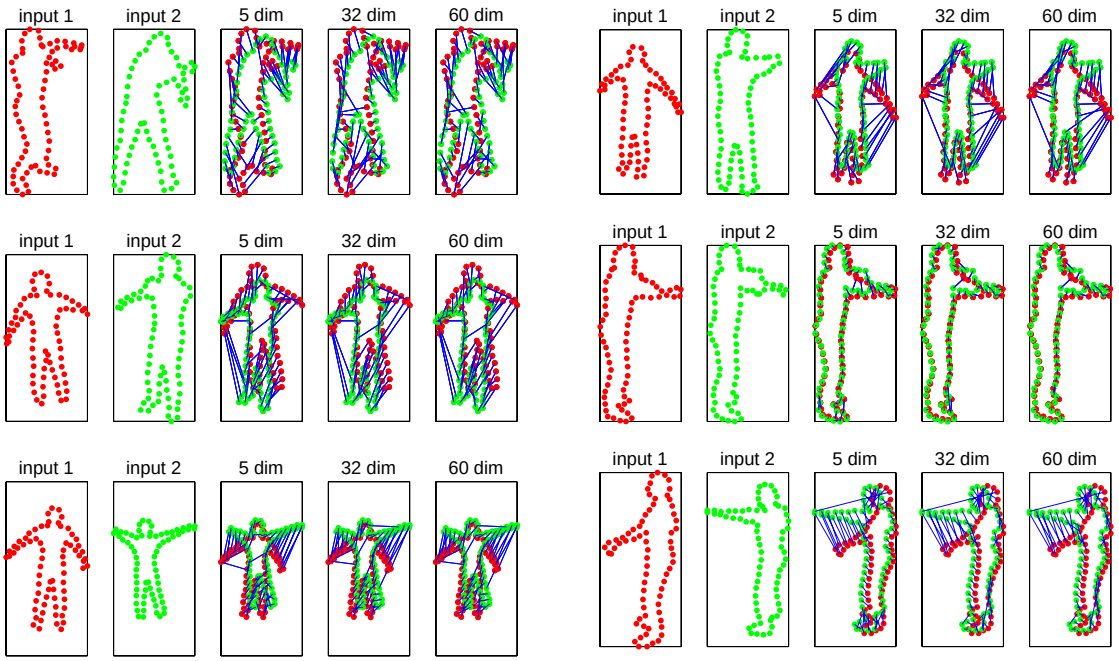


Figure 7-3: A shape context subspace captures local contour variations in much fewer dimensions. The optimal correspondences for feature sets composed of five and 32 shape context PCA coefficients are shown here and compared with the correspondences that result with the raw (d=60) shape context feature. The optimal correspondence found by the raw high-dimensional feature is generally achieved by projections onto a much lower-dimensional subspace.

Locality-Sensitive Hashing Parameters

I constructed a separate set of hash functions for each image data set in order to perform LSH approximate nearest-neighbor retrievals. I determined the LSH parameters (the number of hash tables and the number of bits per hash value) based on the proof in [55] which shows how to select parameters for the case of the Hamming space over a non-binary alphabet, such that the desired level of approximation versus speed tradeoff is achieved. The radius r denotes the distance from a query point where similar points (its "r-neighbors") are believed to lie. The probability of collision of similar examples (examples within distance r), p_1, is set to $1 - \frac{r}{d'}$, where $d' = Zd$, d is the dimension of the embedded vectors, and Z is the largest coordinate value in any of the embeddings. The probability of collision for dissimilar examples (examples farther than $(1+\epsilon)r$ from each other), p_2, is set to $1 - \frac{r(1+\epsilon)}{d'}$. Then the number of bits k and the number of hash tables l are set as follows:

$$k = \log_{1/p_2}\left(\frac{N}{B}\right), \quad l = \left(\frac{N}{B}\right)^{\frac{\log 1/p_1}{\log 1/p_2}}, \tag{7.1}$$

where N is the total number of examples in the database, and B is the maximum number of examples we wish to search exhaustively after hashing. For the complete data set of 136,500 human figure examples, this meant using eight tables and 120-bit functions. For the MNIST

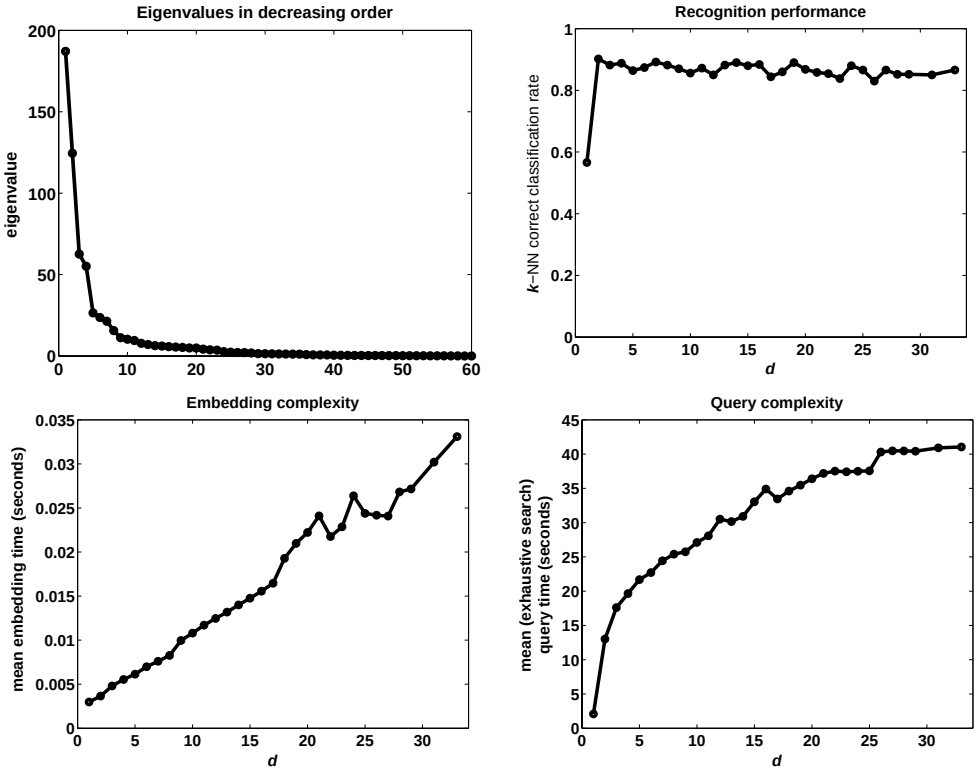


Figure 7-4: Example of using recognition performance to select the feature subspace dimension (d). These data are from the MNIST digits data set. A reserved portion (500 examples) of the training set was used to query the remaining 50,500 in order to optimize d for testing on the true test set. The top left graph shows the magnitudes of the eigenvalues for the PCA subspace over the shape context feature. The top right graph shows the k-NN classification performance that results when the 500 reserved training examples are used as queries, for projection dimensions varying from one to 34 ($k = 3$). The bottom left graph shows the corresponding mean embedding times for the same queries, and the bottom right graph shows the mean query times. Note that even though a number of principal components past the first two dimensions have eigenvalues of significant magnitude, the top right plot reveals that these dimensions are superfluous in terms of recognition performance. The data indicate that the choice of $d = 2$ is appropriate for this data set, since there is an excellent balance between recognition performance and embedding/query efficiency at this point.

digit data set, I used five tables and 96-bit functions. The radius r is determined for each data set by examining the distances and class labels corresponding to the nearest neighbors of an embedded point set under L_1. For all experiments, I set ϵ to 1, which means the upper bound on the query time is $O(N^{\frac{1}{2}})$.

7.1.2 Retrieval Quality

To quantitatively measure the quality of the database retrievals our contour matching method yields, we pose a k-NN classification task for each data set. A k-NN query selects the nearest k examples among a database of training examples, and then labels the test point by a majority vote of these samples.

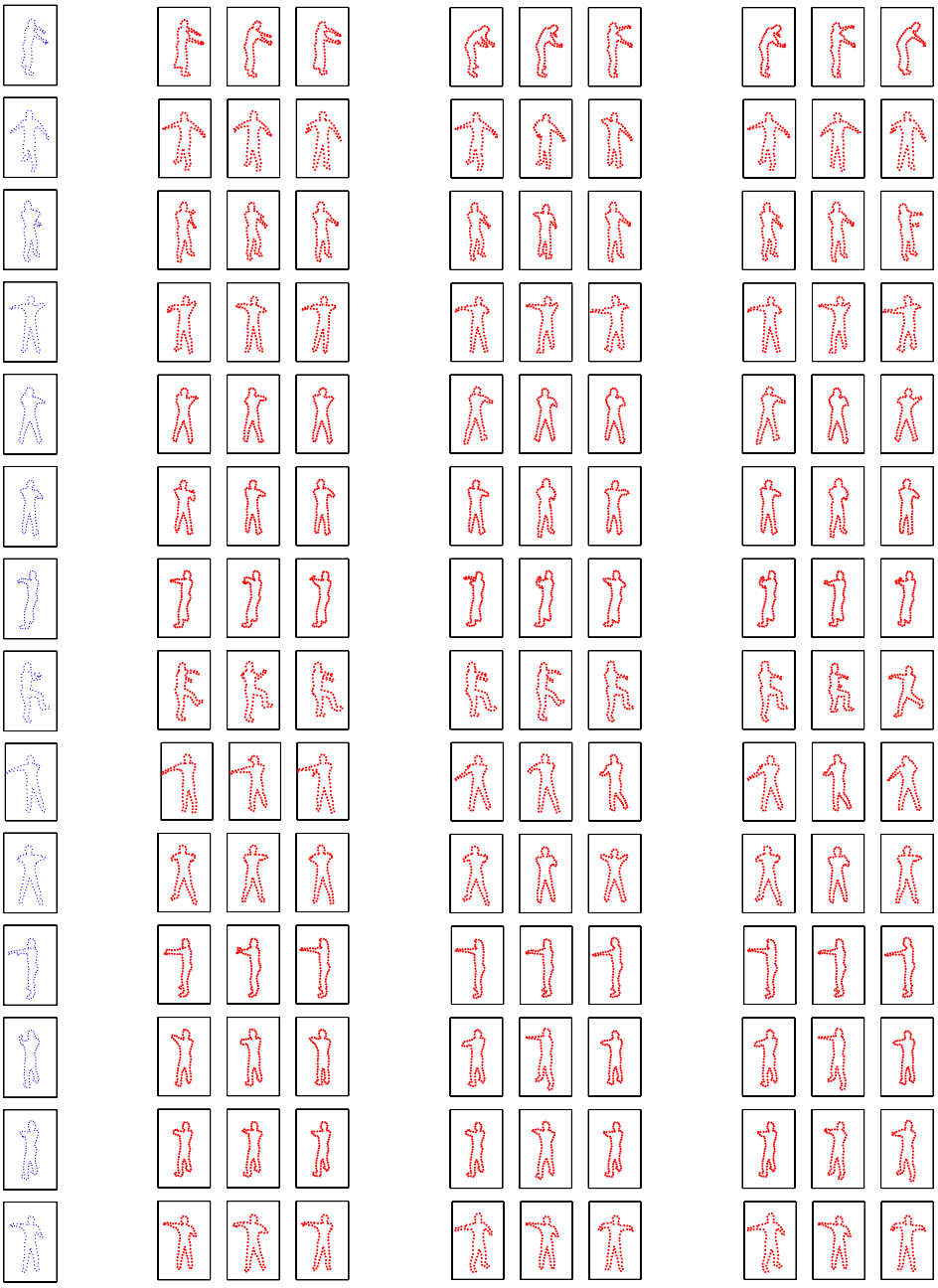
Human Figures Data Set

For the human figures data set, I measure retrieval quality by comparing the 3-D pose of each query example with the pose of the k most similar examples that are retrieved from the database. When the synthetic human figure database is generated, the 3-D pose (a list of 19 3-D joint positions) is recorded for each example. If the joint positions corresponding to a retrieved shape are on average within some threshold of distance from the known joint positions of the query example, we consider the retrieved shape a good match. I chose a threshold of 10 cm, since this is a distance at which the 3-D poses are perceptually similar.

Figure 7-5 shows some example retrievals using our approximate matching approach with synthetic (Poser-generated) query images. Examples of the nearest neighbors that were retrieved for the images from a real person are shown in Figure 7-6. These real image queries contain a single subject performing various actions against a static background. Background subtraction was done automatically using standard frame differencing followed by morphological operations.

Figure 7-7 (a) quantitatively compares the quality of results obtained with the approximate approach with those obtained from EMD (the optimal matching) for a database of 10,000 images. Due to the complexity of EMD, the optimal solutions were necessarily computed with a parallel implementation. In this figure the optimal results were obtained by running the transportation simplex algorithm to compute EMD on full 60D shape context features, whereas results for the two approximations (the embedding and LSH) were obtained using only 2-D shape context subspace features. There is a slight decrease in classification performance at each approximation level; however, I found that the practical bound on the distortion introduced by the L_1 embedding is significantly (about one order of magnitude) lower than the upper theoretical bound. Moreover, the bar graph in Figure 7-7 (a) demonstrates that our method shows only a 4% reduction in accuracy over the exact method, and as I discuss in Section 7.1.3, this small error increase allows us a substantial speedup of four orders of magnitude.

Figure 7-7 (b) shows how well our approach to contour matching serves as a means of estimating 3-D pose from a set of 2-D occluding contour points. The boxplot shows the distributions of the highest rank that is assigned by my method to an example in the database



(a) Queries

(b) Exact EMD

(c) L1 on histogram embeddings

(d) LSH on embeddings

Figure 7-5: An approximate matching retrieves shapes very similar to those retrieved by the optimal matching. This figure shows examples of the three nearest neighbors (left to right in rank order) retrieved with the L_1 embedding for approximate contour matching (c) and approximate contour matching with LSH (d), compared to the nearest neighbors under exact EMD contour matching (b). Examples shown were chosen randomly from 7000 test set results, and nearest neighbors were retrieved from a database of 136,500 examples. Columns (c) and (d) use the 2-D shape context subspace feature; column (b) uses exact EMD applied to full 60-D shape context features. Note that the approximate match results are qualitatively similar, yet are several orders of magnitude faster to compute.

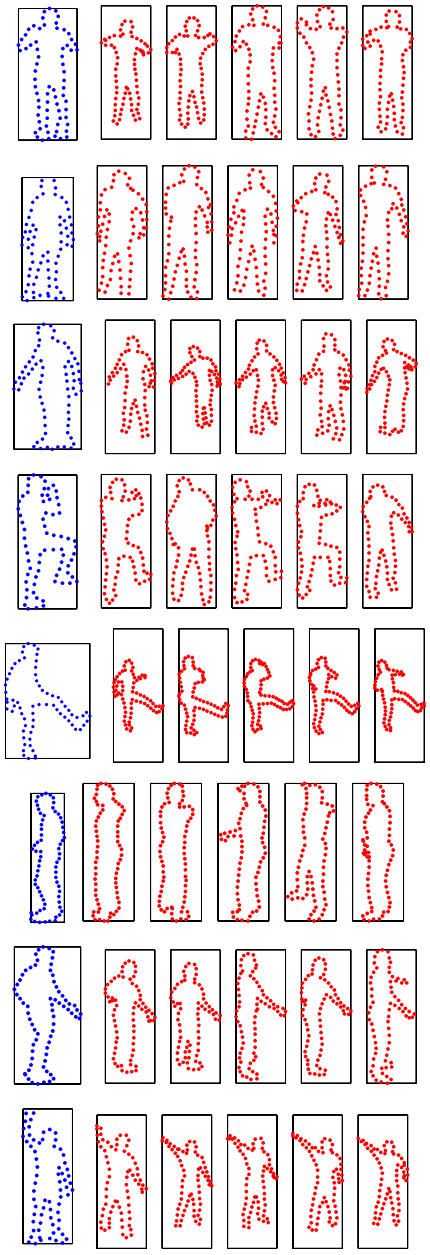
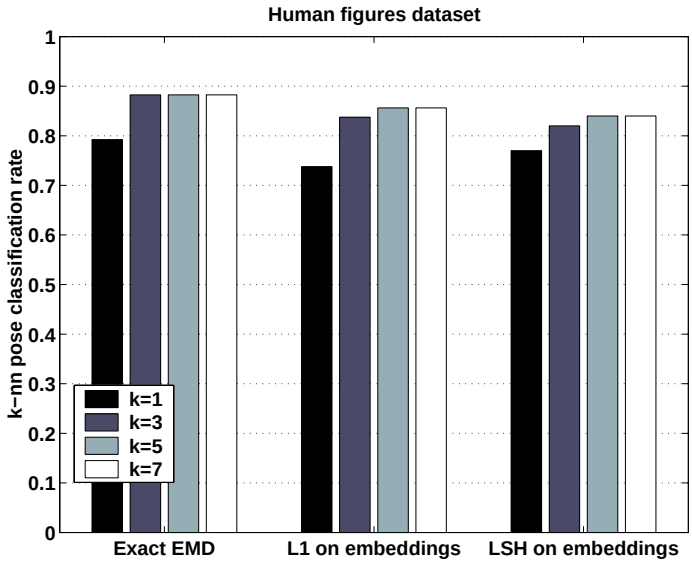
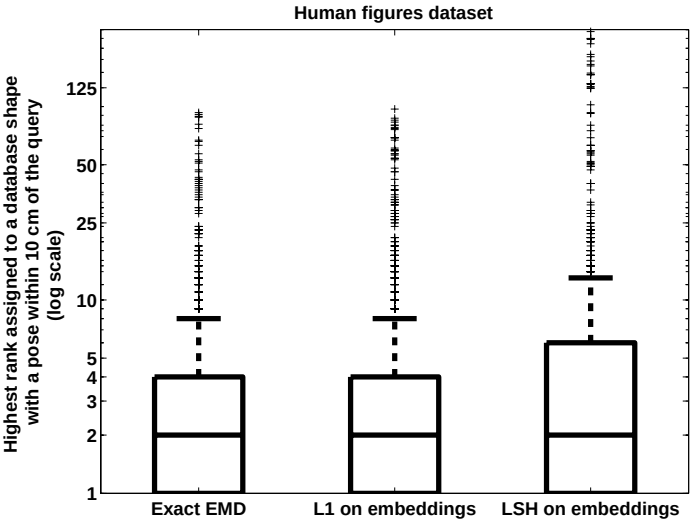


Figure 7-6: Real image queries: examples of query contours from a real person (left image in each group) and the five nearest-neighbor contours retrieved from a synthetic database of 136,500 images using the L_1 distance between the histogram embeddings for 2-D shape context subspace features. The example results shown here were sampled from the 1000 real queries performed.



(a) Accuracy of approximate vs. exact method



(b) Matching performance

Figure 7-7: (a) Comparison of the quality of contour matching retrievals from the exact versus approximate methods for the human figures data set. Quality is measured via a k-NN classification task. The test set and training set are composed of 1000 and 10,000 examples, respectively. For this problem, my approach achieves a speedup of four orders of magnitude over the exact method, at the cost of only a 4% reduction in accuracy. (b) Distributions of the highest rank of a correct match for human figure retrievals. Lines in center of boxes denote median value; the top and bottom of boxes denote upper and lower quartile values, respectively. The dashed lines show extent of rest of the data, and the pluses denote outliers. These boxplots are shown on a log scale. See text for details.

with a very similar (within 10 cm) pose to a query. In most cases, the second nearest neighbor is a "correct" match, meaning it correctly identifies the pose of the query. There are of course ambiguities in the task of inferring 3-D pose from a single frame silhouette view; due to self-occlusions or ambiguities about the orientation of the person, the information provided by the outer contour may be an insufficient indicator of 3-D pose. Nevertheless, we are encouraged by the fact that the correct matches in pose are generally among the highest ranked examples according to our method. We have also investigated how example-based pose estimates may be integrated and improved by considering sequences of silhouette frames [48] or incorporated within a probabilistic model-fitting technique for pose tracking [29].

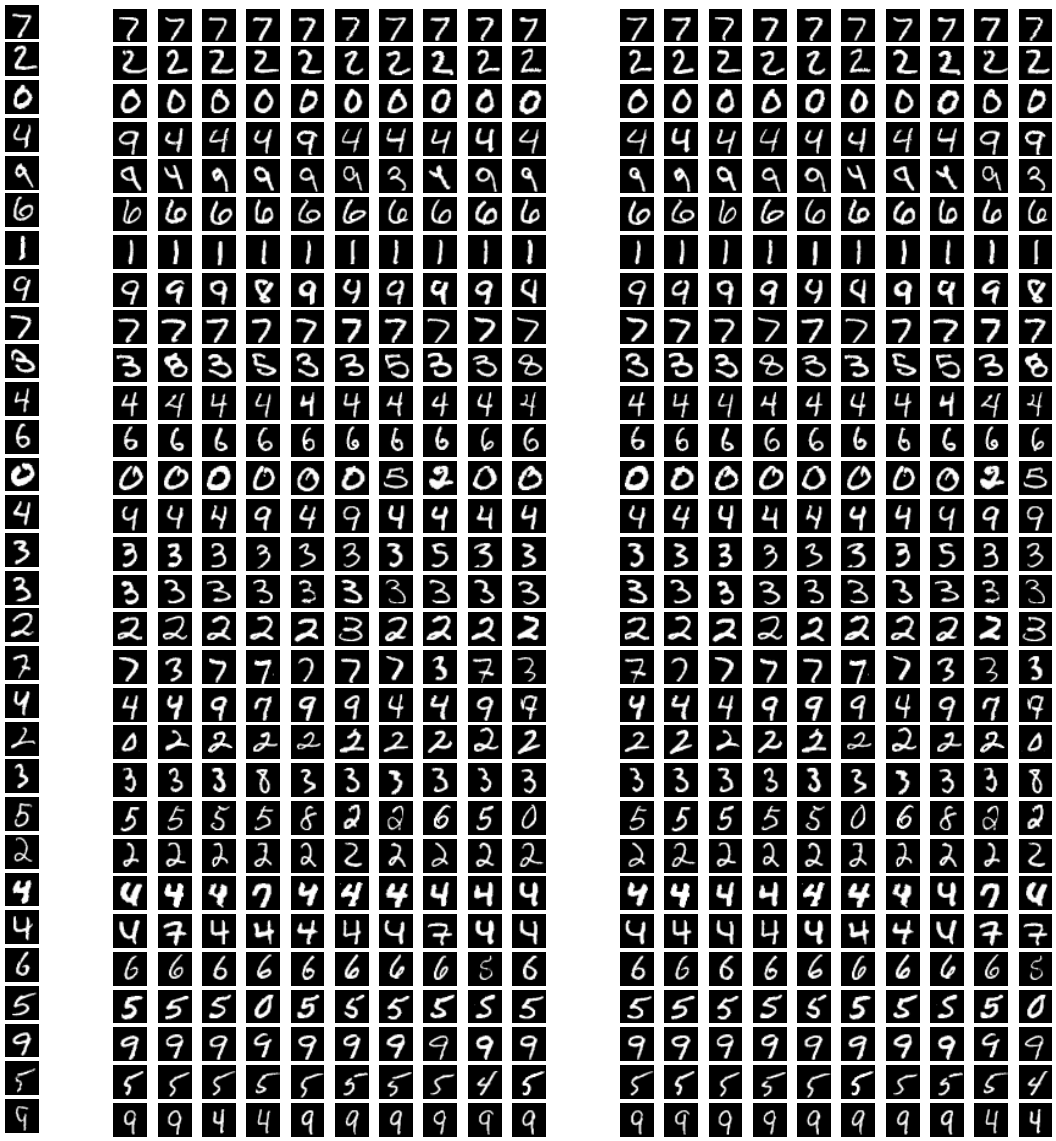
Handwritten Digits Data Set

In a practical system, classification rates can be improved by adding a refining step: some number of the best matches retrieved by the fast approximate method may be fed as input to a more elaborate (and presumably more expensive) matching algorithm for re-ranking. For the MNIST handwritten digits data set, I took this refinement approach. The goal is to correctly label the identity of a query handwritten digit based on the majority vote of its k-NN in the training set of 60,000 examples. First, we use our approximate shape matching method with shape context subspace features to retrieve an initial set of w nearest neighbors for a query. Then we compute the exact shape context distance [7] between this initial nearest neighbor set and the query example, which yields a re-ranking of the neighbors.[3] The k closest neighbors in this re-ranked or refined set are then used to do the classification. Figure 7-8 shows some example matching results for the handwritten digits and illustrates the impact of the refinement stage.

Figure 7-9 shows classification performance for the handwritten digits data set. Since the MNIST digit database is a publicly available benchmark data set, I am able to compare our algorithm's performance against that of many other researchers. (See [80] for a summary of different algorithms' results.) At this writing, the best published classification performance was achieved by the shape context matching method of Belongie et al. [7], which correctly classifies 99.37% of the 10,000-item test set. By combining our approximate contour matching technique with shape context matching as described above, we are able to correctly classify 99.35% of the test set. Note that while the technique in [7] compares query shapes against every database item, with the refinement framework we need only evaluate the shape context distance between the query and a fraction of the database; so with $w = 6000$, our method misclassifies only 65 of the 10,000 test digits, but classification is an order of magnitude more efficient than doing a linear scan of the database.

Even with a much shorter refinement step that evaluates only 50 exact shape context distances, our method correctly classifies 97.42% of the test set. With only 100 exact shape context distances, it correctly classifies 98.08%. In comparison, in [120] Zhang and Malik report a correct classification rate of 97.0% on the same data using their discriminative

[3] Note that the exact shape context distance is different from an optimal matching between features. It first solves the optimal assignment problem, and then given those correspondences, it estimates the thin-plate spline transformation that best aligns the two shapes. The scalar measure of dissimilarity is then the sum of the matching errors between the corresponding points, plus the magnitude of the aligning transform [7].



(a) Queries

(b) Nearest neighbors: initial rankings

(c) Nearest neighbors: refined rankings

Figure 7-8: Example retrieval results for the MNIST handwritten digits database. The approximate matching efficiently filters through the N-item database to find an initial set of w examples whose edge point histograms have low-cost correspondences with the query's histograms. Then, in order to refine the initial rankings, a more expensive shape matching method (such as shape context matching [7]) can be applied to only the w examples that were retrieved with an inexpensive approximate measure. Classification is then done using only the first k re-ranked examples ($k < w$). Each row shows one example. Part (a) shows some query digits from the MNIST test set. Part (b) shows the first $w = 10$ nearest neighbors for these queries under the approximate matching with L_1 (in rank order). Part (c) shows the same 10 nearest neighbors after they are re-ranked by the shape context distance. The first $k = 3$ refined nearest neighbors are used to classify the query digit.

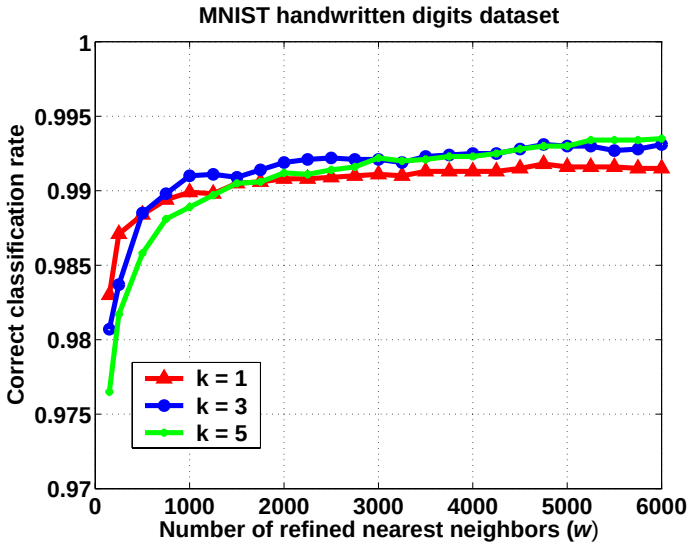


Figure 7-9: Contour matching-based classification of the MNIST handwritten digit test set. Given a query image, the approximate matching between sets of contour features is used to obtain an initial set of nearest neighbors in the training set, and then shape context matching is used to refine the ranks of this set of neighbors. Classification is based on the refined rankings.

classifier, when allowing the same computational cost of 100 exact shape context distances against selected prototypes. In addition, our method's training stage is substantially more efficient and perhaps more easily updatable than that given in [120]. Since the method in [120] relies on k-medoids to determine prototypes from the training set (an $O(NdkT)$ operation for T iterations), it is not clear that it will be able to accommodate new training examples without an expensive re-training step. In contrast, since the "training" of our method consists only of embedding available training examples, additional examples may be incorporated simply by embedding the new examples and entering them into the LSH hash tables.

7.1.3 Empirical Measure of Complexity

With the L_1 histogram embedding, the computational complexity of retrieving the approximate minimum cost feature correspondences for feature sets of cardinality m and dimension d residing in a space of diameter D is $O(md \log D)$. The diameters of the spaces in which our point sets reside are on the order of 10^3 up to 10^5, depending on the representation; with m on the order of 10^2, $d = 2$, theoretically this means that a single embedding and L_1 distance cost requires on the order of 10^3 operations. This is the cost of embedding two point sets, plus performing an L_1 distance on the very sparse vectors.

In practice, with $d = 2$ and $m = 200$, an unoptimized C implementation of our method takes about 0.005 seconds to perform a single matching between the human figure images with exact L_1 (less than 0.005 seconds to compute the two embeddings, plus 7.9×10^{-5} seconds to compute the L_1 distance). In comparison, to compute a single exact EMD distance using

the C implementation of the simplex algorithm provided by Rubner [92] required on average 0.9 seconds for data of the same dimensions. In accordance with the upper bound on the embedding's space requirements, the number of nonzero entries in the sparse embedded vectors was on average 350 for the human figure contours with the shape context subspace representation. For the MNIST digits data, with $d = 2$ and $m = 100$, the average time needed to compute an embedding was 0.003 seconds, and each L_1 distance required 1.7×10^{-4} seconds.[4]

Figure 7-10 (a) gives a summary of the empirical run-time behavior of the embedding. My experiments confirm that the run-time for embedding point sets increases only linearly with the size of the input's dimension or cardinality. This means that the method scales well to handle inputs with large representation sizes.

An even larger payoff for using the approximate matching, however, comes when we use LSH to query a large database with an embedded point set. With ϵ set to 1, the upper bound on the query time is $O(N^{\frac{1}{2}})$. The input must be compared against only a small fraction of the database's embedded vectors—those in the union of the hash buckets that are indexed by the query's embedding.

On average, in my experiments LSH needed to compute only 1915 L_1 distances per query for the human figures database, less than 2% of the database. The median query time for the complete 136,500-item Poser database was only 1.56 seconds. In comparison, a single query with the exact EMD method would require 34 hours (performing a worst-case linear scan of the database). Figure 7-10 (b) shows how query time for the human figure data set varies with the database size.

For the MNIST digits data set, the median query time using LSH (and no refinement step) was 0.14 seconds, and on average a query's embedding was compared to 94 training examples, less than 1% of the database. When the refinement step described in subsection 7.1.2 is included, the query time increases by $w \times g$, where w is the number of rankings that are refined and g is the time needed to perform one shape context distance. (The implementation of the shape context distance that we are using requires $g = 0.07$ seconds.)

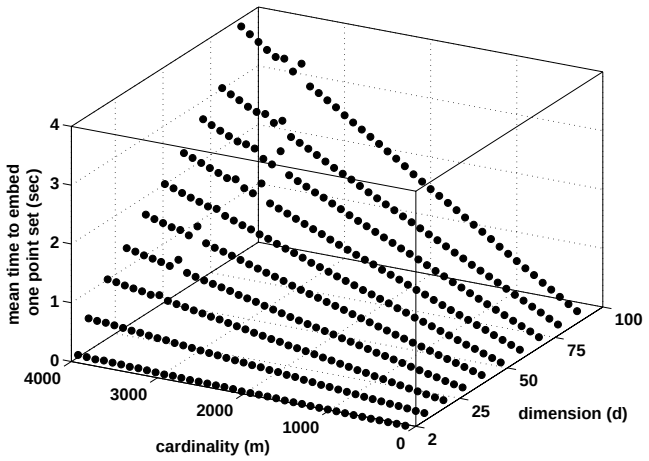
7.2 Image Matching with Sets of Invariant Local Features

In this section, I show results for content-based image matching where images are described by local appearance features. I experiment with three types of image data: scenes, object categories, and textures. For each data set, I directly compare the matching results against the primary alternative techniques based on voting [73, 77, 105, 95] and a bag-of-prototypes approach [102, 26].[5]

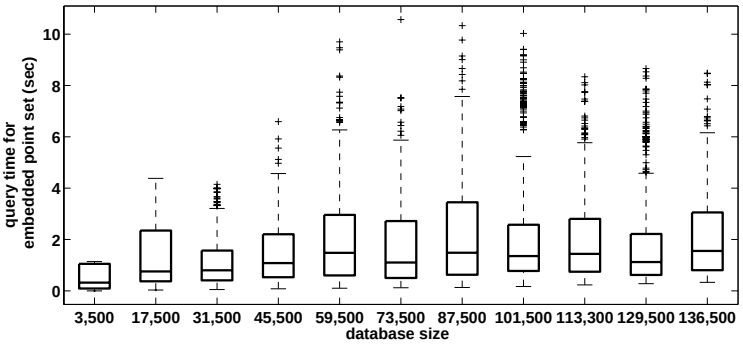
As reviewed in Chapter 2, image features that are stable across varying scales, rotations, illuminations, or viewpoints are desirable for recognition and indexing tasks, since objects

[4] All times reported above are taken from experiments run on a machine with a 1.67 GHz processor and 3.5 GB of memory. Specific run times provided throughout should be considered relative, as recent implementations perform even faster.

[5] I first reported the results in this section in [42].



(a) Histogram computation time



(b) Query time

Figure 7-10: (a) Mean time required to compute the multi-resolution histograms for each of 500 point sets, for varying dimensions and cardinalities. (b) Query time distributions for embedded point sets for increasing database sizes. The test set is composed of 1000 examples from the human figure database, $d = 2$, $m = 200$. The lines in the center of the boxes denote median values; the top and bottom of the boxes denote upper and lower quartile values, respectively. The dashed lines show the extent of the rest of the data, and the pluses denote outliers. The median query time for the 136,500 item database is only 1.56 seconds; exact EMD could require over a day to perform the same query.

are likely to repeat these invariant features in real-world imaging conditions. An interest operator is generally applied to the image to detect stable or distinctive points, and then a local descriptor is extracted from the patch or ellipse around each interest point. I represent each grayscale image by the unordered set of the local descriptor vectors extracted from its interest points.

In these experiments I use two types of interest operators: the Harris-Affine detector described in [79], which detects points that are invariant to scale and affine transformations, and the Scale Invariant Feature Transform (SIFT) interest operator of Lowe [73], which detects points that are invariant to scaling and rotation and has been shown in practice to be resistant to common image transformations. I employ the low-dimensional gradient-based descriptor called PCA-SIFT [63] to describe patches extracted at these interest points. Other interest operators or descriptors are of course possible. For these experiments, the weight given to each feature in a set is normalized to produce equal-mass sets. At the time when I ran these experiments for [42] (which pre-dated my development of the partial match for variable-sized sets) the weight normalization was necessary to fit the equal-mass requirement of the L_1 embedding of [57]. See Chapter 8 for results using my approximate partial matching between unequal-mass sets.

The type of local features used may not individually contain explicit spatial information. However, my method allows spatial constraints among the local features to be enforced by augmenting the feature representation to include an encoding of the geometry of other interest points in relation to each given feature. The descriptor for each interest point is concatenated with invariant information about the configuration of its spatially nearest interest points in the image. Then, when these higher-order feature distributions are compared under the set matching, the low-cost feature matching seeks to satisfy the additional constraints.

There are various possible constraints to include. To designate simple proximity constraints between features, each feature $\mathbf{x}^{\mathbf{p}_j}$ is paired with its n nearest-located features in the image to produce n new features of the form $[\mathbf{x}^{\mathbf{p}_j}, \mathbf{x}^{\mathbf{p}_i}]$, where $\mathbf{p}_i$ is the ith closest interest point to $\mathbf{p}_j$, for $1 \leq i \leq n$. Additionally, the angle of separation θ_i between two features' dominant orientations can be incorporated to further constrain their geometric relationship, creating features of the form $[\mathbf{x}^{\mathbf{p}_j}, \mathbf{x}^{\mathbf{p}_i}, \theta_i]$. Both result in a similarity-invariant descriptor, since the length ratios of two lines and the angle between two lines are invariant binary relations under similarity transformations [35]. Other constraints based on affine or projective invariants are possible but would require larger tuples.

7.2.1 Methodology

For each data set, I use the *normalized average rank* $\bar{r}$ described in Chapter 6 (and repeated here for convenience) as a measure of matching performance:

$$\bar{r} = \frac{1}{NN_R}\left(\sum_{i=1}^{N_R} r_i - \frac{N_R(N_R-1)}{2}\right), \tag{7.2}$$

where r_i is the rank at which the ith relevant image is retrieved, N_R is the number of relevant images for a given query, and N is the number of examples in the database. The normalized rank is 0 for perfect performance (i.e., when all relevant images in the database are retrieved as a query's nearest neighbors), and it approaches 1 as performance worsens; a random retrieval results in an expected normalized rank of 0.5 [82]. I also report results in terms of the classification error rates when the k-nearest neighbors (k-NN) are used to vote on the query's class label, with $k = 3$ in all experiments. The normalized rank is more comprehensive, but recognition error is sometimes a more intuitive measure of performance.

For each data set, I compare our algorithm's performance with two other techniques: a voting technique and a prototypical-feature technique. All three methods share the idea of representing images based on their sparse sets of invariant features, but they vary in the way that they judge similarity between the feature sets.

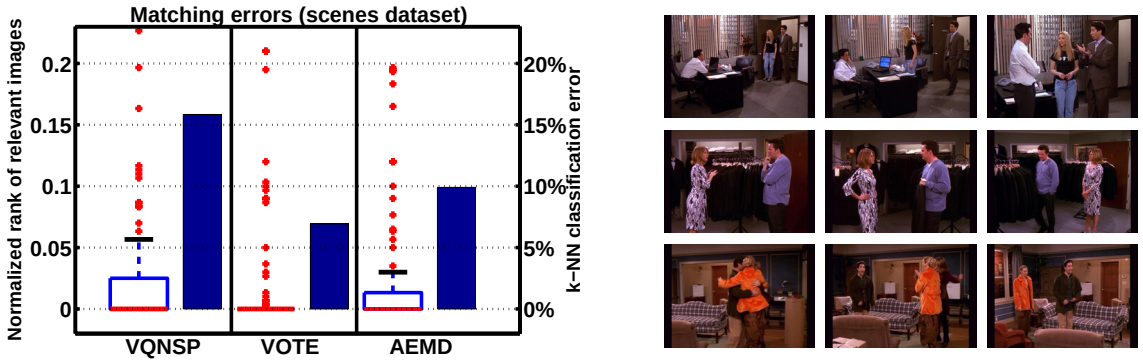
For the voting scheme, each feature in a query image is compared against all of the features extracted from database images, and then that query feature casts a vote for the database image containing the nearest neighbor feature in terms of L_2 distance. The database images are then ranked in similarity to the query based on the number of votes they have received. Note that when measuring the voting technique's performance, I used an exact (exhaustive) search to determine each feature's nearest neighbor, but exhaustive search is computationally infeasible in practice. So the voting results should be considered an upper bound; in practice, an approximate-NN technique such as LSH or BBF [73] is used to make voting computationally tractable, but at some cost of matching error.

For the prototypical feature scheme, vector quantization is used to map all descriptors to a discrete set of prototypes, which are found by running k-means on a set of examples containing images from each class. Each image is represented by a vector giving the frequency of occurrence of each prototype, weighted by the *term frequency - inverse document frequency*. The database images are then ranked in similarity to the query based on the normalized scalar product between their frequency vectors. Our implementation is modeled on the video data mining method of Sivic and Zisserman [102]; we omit the "stop-list" and temporal feature tracking steps since we are matching static, non-sequential images in addition to video frames in these experiments.

To extract the SIFT, PCA-SIFT, and Harris-Affine features in these experiments, I used the code that the authors of [73, 63, 79] have provided online. I used the first eight dimensions of the PCA-SIFT features as input to all methods, and on the order of 10^2 prototypes for the prototypical-feature method (100, 400, and 700 clusters for the scenes, objects, and textures, respectively); these parameters were optimized for recognition performance on a held out set of examples.

7.2.2 Scene Matching

Shot matching is a specific use of scene recognition where the goal is to automatically identify which video frames belong to the same scene in a film. To test our method in this regard, I used a data set of images from six episodes of the sitcom *Friends*. I extracted one frame for every second of the video (so as to avoid redundancy in the database), for a total



(a) Retrieval errors

(b) Example retrievals

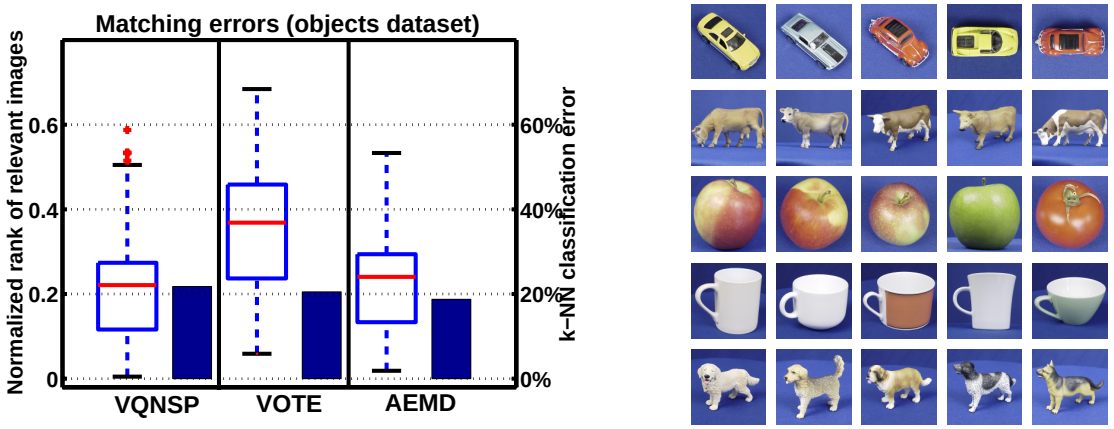
Figure 7-11: Matching results for the scenes data set. The plot on the left (a) shows the distribution of normalized ranks of relevant images (boxplots shown for each method, axis on left) as well as the k-NN classification error rates for $k = 3$ (dark bar on right for each method, axis on right). VQNSP denotes matching with the normalized scalar product applied to vector-quantized features, VOTE denotes voting with each feature independently, and AEMD denotes matching with approximate correspondences on raw feature distributions. A normalized rank of zero signifies a perfect ranking, where all relevant images are returned as the closest nearest neighbors. The red lines in the boxes denote median values; the top and bottom of boxes denote upper and lower quartile values, respectively. Dashed lines show the extent of the rest of the data; pluses denote outliers. The images next to the plot are example retrievals obtained with the approximate matching. The query in each row is the leftmost image, and the subsequent images are its nearest neighbors.

of 8,335 images. The SIFT interest operator was used to detect keypoints, and PCA-SIFT descriptors formed the feature sets.

With the approximate-NN technique LSH, it is only necessary to compute L_1 distances between the query's histogram embedding and a small portion of the database embeddings. In this case, queries on average required only 480 distances to be computed, i.e., on average each image was compared to 5% of the database.

Figure 7-11 shows the matching performance of my method, voting, and prototype-feature matching on a ground truth subset of the *Friends* data set containing 100 images that were hand-labeled with scene identity. These 100 images contain frames from 27 different scenes, with about four images from each scene. Each image from the same scene is taken from a different camera shot so that the viewpoints and image content (actors' positions, etc.) vary. I used leave-one-out cross validation (LOOCV) for these ground truth tests in order to maximize the use of the labeled data.

Using the k-NN under each method as a classifier of scene identity, voting classifies 93% correctly, my approach classifies 90% correctly, and the VQ approach classifies 84% correctly. This experiment indicates that the salient SIFT features were reliably extracted in each instance of a scene, making voting very successful. This seems reasonable; although the images have some viewpoint variation, due to the nature of the source—a TV sitcom



(a) Retrieval errors

(b) Example retrievals

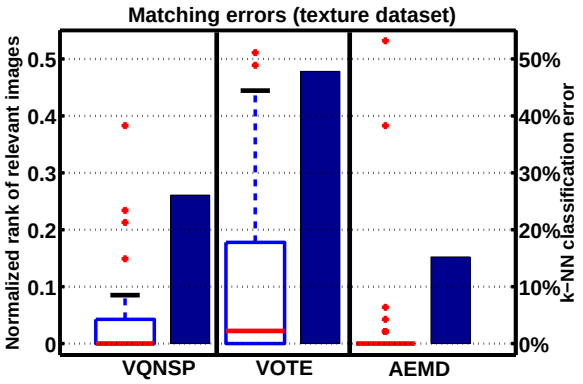
Figure 7-12: Matching results for the objects data set. Plotted as in previous figure.

set—they have consistent quality and illumination, and each scene is unique enough that discriminating features have good leverage under voting. However, this voting result did require exhaustive search for NN features, which is computationally prohibitive in practice. We would expect marginally worse performance from voting if an approximate method were used to find NN features, as the reduction in computational complexity does come at the cost of some accuracy. My method does nearly as well as "perfect" voting, yet is much more efficient; exact voting requires hours for a match that my method performs in under a second (see Section 7.2.5). The relevant rank distribution is somewhat wider under the prototype-feature method (VQNSP), indicating that the quantization of the features adversely affects performance for this data set.

7.2.3 Object Category Retrieval

Next I evaluated our method on an object retrieval task using the ETH-80 database [69], which contains images of 80 objects from eight different classes in various poses against a simple background. I included five widely separated views of each object in the database, for a total of 400 images. The Harris-Affine detector was used to detect interest points, and PCA-SIFT descriptors were again used to compose the feature sets. The k-NN classification accuracy and the relevant rankings were measured via cross-validation, where all five views of the current test object were held out from the database. Since no instances of the query object are ever present in the database, this is a more challenging task than the previous scene recognition experiment; the goal is to rank all other instances of the object class as a query's nearest neighbors, but the other instances will exhibit intra-class appearance variation.

Figure 7-12 shows the matching performance of the three methods for this data set. Our approximate feature matching approach (AEMD) gives the best k-NN classification accuracy, meaning that the three nearest neighbors it found for each query were most consistently from the correct class, although by this metric the other methods do nearly as well (errors



(a) Retrieval errors

(b) Example retrievals

Figure 7-13: Matching results for the textures data set. Plotted as in previous figures.

range from 19% to 22%). VQNSP achieves a slightly lower median relevant ranking score than AEMD. However, both AEMD and VQNSP assign substantially better relevant rankings than VOTE; under AEMD and VQNSP many of the relevant images from the same object class were usually given high rankings, whereas VOTE could only assign high ranks to a few very similar objects.

I found local gradient-based descriptors to be fairly effective for matching these images; however, this representation does have some limitations that are revealed by this database. Some objects from different classes are similar enough in terms of shape and local corner-based interest points that the PCA-SIFT feature cannot discriminate between them. For instance, an apple query at times retrieves a tomato among its nearest neighbors (see Figure 7-12, third row of images). Additional features such as color distributions may be necessary to improve performance for any of the methods on this type of data.

Voting does poorly in this experiment because it finds matches based on how well individual features correspond, ignoring the higher-level information captured by the distribution of local features that occur on an object. The appearance variation across different instances of the same object class cause variations in the detected local features, which misguides the independent votes cast by the query object's features. This indicates that generally the set of features an object exhibits can be more discriminative than each individual feature considered separately, causing the distribution-based approaches (AEMD and VQNSP) to be more successful. Thus while voting appears to be an effective but expensive strategy when searching for the same instance of an object (as in the scene matching task), this data set shows it to be a weaker approach for category-level matching tasks.

7.2.4 Texture Matching

Finally, I have experimented with a data set of texture images. There are issues unique to comparing textures as opposed to the scenes and objects compared in the above experiments; in particular, textures are often defined in terms of how local features or structures co-

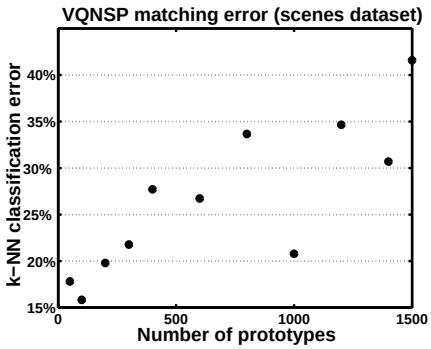


Figure 7-14: The matching performance of prototype-based methods can be sensitive to the number of prototypes chosen.

occur. Each instance of a texture is a sample of an underlying, nonuniform pattern. This makes texture matching another domain that is especially amenable to my method since it captures the joint statistics of invariant features. I ran experiments with the publicly available VisTex Reference database, which contains 168 images of textures under real-world conditions, including both frontal and oblique perspectives and non-studio lighting.

Figure 7-13 shows the matching performance of the three methods for this data set. I randomly selected a set of 25 VisTex examples.[6] Each image was split into halves, making 50 images, and again we tested with LOOCV. The goal in this test was for each query to match most closely with the other half of the texture image from which it originated. Note that most of the textures are mainly homogeneous at a high level, but that the two halves of each image still have significant variations, and different structures occur in each half.

Our approximate set matching approach results in the best texture classification performance on this data set. While the methods capturing co-occurrence information (AEMD and VQNSP) assign rankings that are tightly distributed close to zero (the ideal normalized rank), voting fails to consistently assign low ranks to images of the same texture, resulting in a much wider distribution centered at 0.14. While voting was successful for scene matching where distinct features were repeated, it breaks down for texture matching due to the variation of the features within different samples of the same texture. As with the object categorization task, voting suffers because it does not capture co-occurrence information that is critical for texture matching. Based on the texture retrievals obtained by voting, I found that it risks casting excessive votes for images containing a repeated "generic" feature.

7.2.5 Discussion

In all these image retrieval experiments, I found that VQNSP matching quality was sensitive to the clustering that defined the prototypes (see Figure 7-14), as was also observed in [26]. The number of clusters can be thought of as the "bin size" for this method – more clusters means smaller bins. The quality of the matching varied substantially depending on both the

[6]The entire VisTex database was not used for the comparative study due to the computation time needed to get exact NN features for voting. Using all the textures, AEMD achieves a median normalized relevant rank of 0.003.

Figure 7-15: Mobile image-based object search.

number of clusters, as well as the random starting point of k-means. Additionally, the type of images used to create the prototypes affected VQNSP's matching performance. I found it necessary to build prototypes from images that were very similar to the test examples in order to achieve VQNSP's best performance (i.e., other scene images from the same video, or other examples from the same objects in the ETH-80 database), which suggests that it may be difficult to use the prototype-based methods to do "general purpose" matching on purely unseen test data.[7] As shown in Figures 7-11, 7-12, and 7-13, compared to VQNSP, AEMD more consistently ranked both the *Friends* and textures images (and equally ranked the ETH-80 images), and it does not require a parameter choice since it matches the (unbinned) features themselves.

As in the previous shape retrieval experiments, I set ϵ to 1, making the upper bound on the query time $O(s\sqrt{N})$, where s is the number of nonzero histogram entries, $s = O(m \log D)$. In comparison, to process one query, a voting scheme must perform m retrievals from a database with on the order of $N \times m$ items in order to match each of its d-dimensional features to the database, making a single query cost $O(dm^2N)$ if exact NN features are found. For the prototype-frequency method, the query time has an upper bound of $O(dpmN)$, where p is the number of prototypes. To give a concrete example, with our implementations and $d = 8$, $N = 8335$, $m = 1000$, $p = 500$, and $D = 316$, a single voting query requires over 2.2 hours, a single query with the prototype-based method requires 1.62 seconds, and our method requires 0.49 seconds.

7.3 Related Applications

The results in this chapter have focused on the quality of correspondence-based image and shape matching using approximate matching and similarity search techniques for local feature representations. As the experiments are presented, the end goal is to produce highly

[7] However, researchers have made some advances on the problem of efficiently developing good image vocabularies since I performed these experiments. See for instance [60].

efficient, good quality retrievals. However, in work with my colleagues, I have also shown higher-level applications of fast approximate retrieval techniques for situated search from a mobile device (see Figure 7.2.5) and probabilistic pose tracking.

We have applied our approximate matching algorithm in a novel system for image-based situated search on a mobile platform in [117]. Current systems for search are driven by keywords reflecting concepts already known by a user. But often when a user is in an unfamiliar environment or encounters a new object, keywords alone cannot adequately specify what information is desired. When touring a new city, how should a user use her mobile phone or PDA to access a web page describing an unmarked building of interest? When hiking in the wilderness, how could a user get immediate information about an exotic flower or creature he happens upon? Our system allows visual cues to be used in concert with keywords on a mobile device to find web pages containing relevant text and images. With this "point-and-search" interface, a user has a direct way to retrieve pertinent information on the spot.

Chapter 8

Supervised Learning and the Pyramid Match Kernel

The previous chapter focused on content-based retrieval applications using approximate correspondences and randomized hashing, where the approximate matching scores were simply a vehicle to identify similar examples or infer categories via nearest neighbors. In this chapter I show that we can in fact learn from sets of features using the pyramid match as a kernel function and exploiting well-developed kernel-based learning methods. I concentrate on two general supervised learning techniques: discriminative classification and regression.

8.1 Kernel Methods and the Pyramid Match

For both classification and regression I make use of existing kernel-based learning methods, but insert the pyramid match kernel (PMK) as the set kernel with which they operate. In particular, I train Support Vector Machines (SVMs) and Support Vector Regressors (SVRs) to perform classification and regression with the PMK. In this section, for the reader's convenience, I briefly overview binary classification with SVMs, using the notation of [93]. I then discuss a practical issue regarding large diagonals that may occur in pyramid match kernel matrices.

8.1.1 Support Vector Machines and Regressors

Consider the problem of learning to predict binary labels for patterns in the input space $X \subseteq \Re^N$; we want to learn a function $f : X \rightarrow \{\pm 1\}$ based on some training data pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l) \in \Re^N \times \{\pm 1\}$. When given a new unseen example $\mathbf{x}$, this function should predict the correct label y for examples generated from the same underlying distribution as the training data, that is, $f(\mathbf{x}) = y$.

A hyperplane classifier $(\mathbf{w} \cdot \mathbf{x}) + b = 0$, $\mathbf{w} \in \Re^N, b \in \Re$ can be considered as a decision

function: $f(\mathbf{x}) = \text{sgn}((\mathbf{w} \cdot \mathbf{x}) + b)$. Among all such hyperplanes separating the training data into two classes, the optimal separating hyperplane is the one yielding the maximum margin of separation between the classes, as it provides the lowest bound on the expected generalization error. To construct the optimal hyperplane, the following optimization problem is solved:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} & y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, l. \end{array} \tag{8.1}$$

The constrained optimization problem is solved by introducing Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i (y_i \cdot ((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1). \tag{8.2}$$

The hyperplane decision function is then

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i \cdot (\mathbf{x} \cdot \mathbf{x}_i) + b\right), \tag{8.3}$$

and those patterns from the training set $\mathbf{x}_i$ whose corresponding α_i is non-zero in the solution are called the *support vectors*. Only the support vectors, which lie on the margin, play a role in the optimization and in the classification of a novel example. (For cases where the classes are non-separable, slack variables may be introduced via an upper bound C on the Lagrangian multipliers α_i.)

To construct Support Vector Machines, the optimal hyperplane is computed in a feature space: $\Phi(\mathbf{x}_i)$ is substituted for each training example $\mathbf{x}_i$. Since all patterns occur only in dot products in the decision function, a Mercer kernel K can be inserted, making it unnecessary to compute an explicit mapping to the feature space:

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i \cdot (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)) + b\right) \\ &= \text{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i \cdot K(\mathbf{x}, \mathbf{x}_i) + b\right). \end{aligned} \tag{8.4}$$

The associated quadratic program is then

$$\begin{array}{ll} \text{maximize} & W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} & \alpha_i \geq 0, \quad i = 1, \ldots, l, \text{and } \sum_{i=1}^{l} \alpha_i y_i = 0. \end{array} \tag{8.5}$$

As we can see above, relative to some particular training data, only the matrix of kernel values between all pairs of training examples ($K(\mathbf{x}_i, \mathbf{x}_j)$, for $i, j = 1, \ldots, l$) is needed to train an SVM. Therefore, in general the inputs $\mathbf{x}_i$, $\mathbf{x}_j$ need not be vectors; the input space X need not be in $\Re^N$, but could for instance consist of graphs, trees, strings, or sets, provided an appropriate Mercer kernel is available. The kernel's similarity values determine

the examples' relative positions in an embedded space, and quadratic programming (QP) is used to find the optimal separating hyperplane between the two classes in this space (for classification), or to fit a function over the examples (for regression). Note that while linear relations are sought in the embedded (feature) space, a decision boundary may still be non-linear in the input space, depending on the choice of a feature mapping function Φ. Because the pyramid match kernel is positive-definite (see Section 3.4) we are guaranteed to find a unique optimal solution to the QP problem.

8.1.2 Dealing with Large Kernel Matrix Diagonals

The pyramid match kernel can produce kernel matrices with dominant diagonals, particularly as the dimension of the features in the sets increases. The reason for this is that as the dimension of the points increases, there are a greater number of finer-resolution histogram levels in the pyramid where two input sets will have few shared bins. Once the quantization level is coarse enough, the two sets will start having significant histogram intersection values. However, these intersections will receive lower weights due to the $\frac{1}{d2^i}$ weighting scheme, which by construction accurately reflects the maximal distance between points falling into the same bin at level i. On the other hand, an input set compared against itself will result in large histogram intersection values at each level—specifically intersection values equal to the number of points in the set, which after normalization generates a diagonal entry of one.

The danger of having a kernel matrix diagonal that is significantly larger than the off-diagonal entries is that the examples appear nearly orthogonal in the feature space, in some cases causing an SVM or SVR to essentially "memorize" the data, and thus impairing its sparsity and generalization ability [99]. Nonetheless, we are able to work around this issue by modifying the initial kernel matrix in such a way that reduces its dynamic range, while preserving its positive-definiteness. I use the functional calculus transformation suggested in [112]: a subpolynomial kernel is applied to the original kernel values, followed by an empirical kernel mapping that embeds the distance measure into a feature space.

When necessary to reduce diagonal dominance, first kernel values K_{ij} generated by the PMK are updated to $K_{ij} \leftarrow (K_{ij})^p$, $0 < p < 1$. Then the kernel matrix $\mathbf{K}$ is replaced with $\mathbf{K}\mathbf{K}^{\mathrm{T}}$ to obtain the empirical feature map $\Phi_e(\mathbf{y}) = [K(\mathbf{y}, \mathbf{x}_1), \ldots, K(\mathbf{y}, \mathbf{x}_l)]^{\mathrm{T}}$ for l training examples. As in [112], the parameter p may be chosen with cross-validation. Note that this post-processing of the kernel matrix is not always necessary; both the dimension of the points as well as the specific structure of a given data set will determine how dominant the initial kernel matrix diagonal is. In addition, the gradual growth of bins in a vocabulary-guided match make it possible to avoid dominant diagonals.

8.2 Discriminative Category Recognition

In this section I report on object recognition experiments using SVMs and provide baseline comparisons to other methods. I use the SVM implementation provided in the LIBSVM library [17] and train one-versus-all classifiers in order to perform multi-class classification.

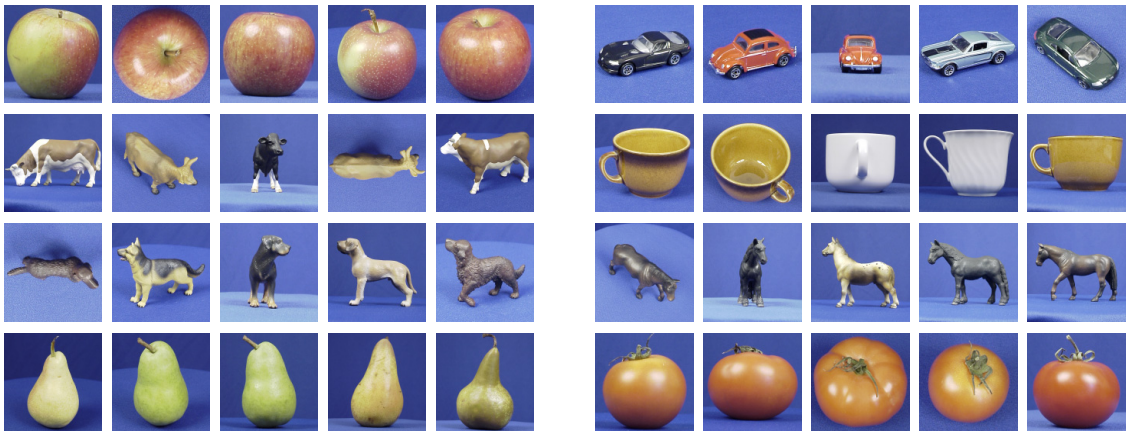


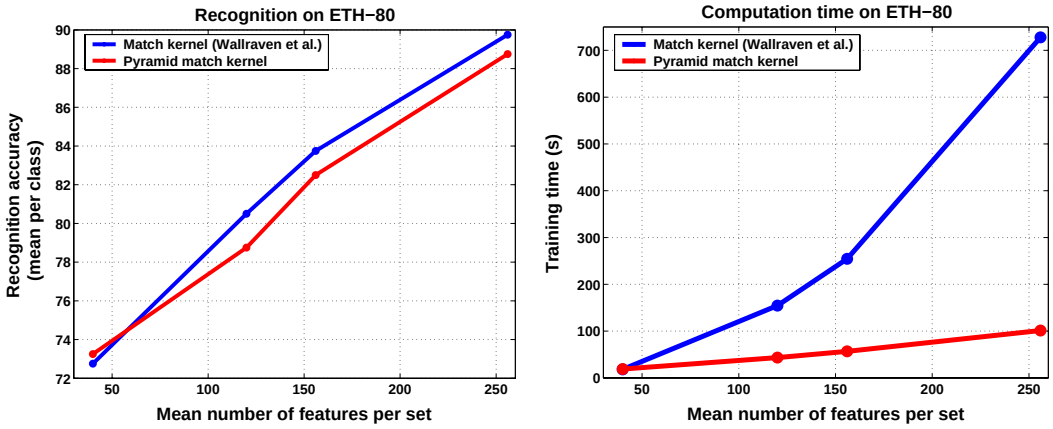
Figure 8-1: Example images from the ETH-80 objects database. Five images from each of the eight object classes (apple, cow, dog, pear, car, cup, horse, and tomato) are shown here.

Local affine- or scale-invariant feature descriptors extracted from a sparse set of interest points in an image have been shown to be an effective, compact representation (e.g. [73, 77]). This is a good context in which to test the pyramid match kernel, since such local features have no inherent ordering, and it is expected that the number of features will vary across examples. In the following I experiment with two publicly available databases and demonstrate that my method achieves better or comparable object recognition performance at a significantly lower computational cost than other state-of-the-art approaches. All pyramid match run-times reported below include the time needed to compute both the pyramids and the weighted intersections, and were measured on 2.4 GHz processors with 2 GB of memory.

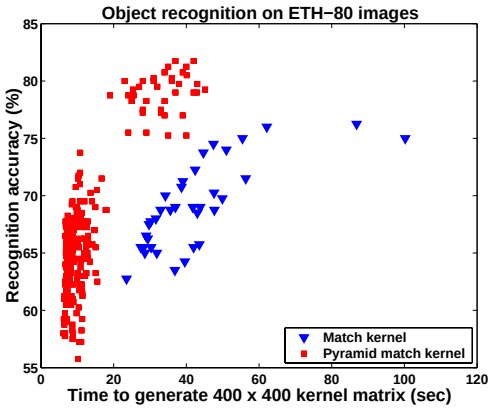
8.2.1 ETH-80 Data Set

A performance evaluation given by Eichhorn and Chapelle [30] compares the set kernels developed by Kondor and Jebara [65], Wolf and Shashua [115], and Wallraven et al. [108] in the context of an object categorization task using images from the publicly available ETH-80 database [69]. The experiment uses eight object classes, with 10 unique objects and five widely separated views of each, for a total of 400 images (see Figure 8-1). A Harris detector is used to find interest points in each image, and various local descriptors (SIFT features of [73], JET, patches) are used to compose the feature sets. A one-versus-all SVM classifier is trained for each kernel type, and performance is measured via cross-validation, where all five views of an object are held out at once. Since no instances of a test object are ever present in the training set, this is a categorization task (as opposed to recognition of the same specific object).

The experiments show the polynomial-time "match kernel" of Wallraven et al. [108] and Bhattacharyya kernel of Kondor and Jebara [65] performing best, with a classification rate of 74% using on average 40 SIFT features per image [30]. Using 120 interest points, the Bhattacharyya kernel achieves 85% accuracy. However, the study also concluded that the cubic complexity of the method made it impractical to use the desired number of features.



(a) Recognition accuracy and training time with input size



(b) Time versus accuracy

Figure 8-2: Both the pyramid match kernel and the match kernel of Wallraven et al. yield comparable recognition accuracy for input sets of the same size ((a), left plot). Both benefit from using richer (larger) image descriptions. However, while the time required to learn object categories with the quadratic-time match kernel grows quickly with the input size, the time required by the linear-time pyramid match remains very efficient ((a), right plot). Allowing the same run-time, the pyramid match kernel (with $T = 1$) produces better recognition rates than the match kernel (b).

I evaluated the pyramid match kernel on this same subset of the ETH-80 database under the conditions provided in [30], and it achieved a recognition rate of 83% using PCA-SIFT features [63] from all Harris-detected interest points (averages 153 points per image) and $T = 8$. Restricting the input sets to an average of 40 interest points yields a recognition rate of 73%. Thus the PMK performs comparably to the others at their best for this data set, but is much more efficient than those tested above, requiring time only linear in the number of features.

In fact, the ability of a kernel to handle large numbers of features can be critical to its success. An interest operator may be tuned to select only the most salient features, but in my experiments I found that recognition rates always benefitted from having larger numbers of features per image with which to judge similarity. Figure 8-2 (a) depicts the run-time versus recognition accuracy of my method as compared to the match kernel of Wallraven et al. [108], which has $O(dm^2)$ complexity. The match kernel computes kernel values by averaging over the distances between every point in one set and its nearest point in the other set. These are results from our own implementation of the match kernel. Each point in the figure represents one experiment; the saliency threshold of the Harris interest operator was adjusted to generate varying numbers of features, thus trading off accuracy versus run-time. With no filtering of the Harris detector output (i.e., with a threshold of 0), there is on average 153 features per set. To extract an average of 256 features per set I used no interest operator and sampled features densely and uniformly from the images. Computing a kernel matrix for the same data is significantly faster with the pyramid match kernel (right plot in Figure 8-2 (a)), and for similar run-times my method produces much better recognition results (Figure 8-2 (b)).

The ETH-80 data set contains relatively simple images against uncluttered backgrounds, making the limit of the number of features we can extract per image rather small. Thus while the pyramid match is significantly less expensive than the match kernel, the match kernel computation is still tractable. However, for more complex images where several thousand descriptors per image may be desirable, this quadratic growth in cost does become infeasible, demanding on the order of weeks to train a classifier that the pyramid match can build in hours.

8.2.2 Caltech-4 Data Set

To evaluate the impact of vocabulary-guided pyramid matching, I ran recognition experiments on a well-studied benchmark database, the Caltech-4 collection of motorcycle, face, car, and airplane images (see Figure 8-3). I trained SVMs with 200 images per class and tested with all the remaining (2388) images. I extracted features using both the Harris and Maximally Stable Extremal Region (MSER) [76] detectors and the 128-D SIFT [73] descriptor. The Harris detector found between 10 and 700 features per image, while the MSER operator detected between 3 and 458. Features from different detectors are matched only to one another (i.e., MSER features to MSER features, and Harris to Harris), producing two kernel values for every set-to-set comparison, which are then summed to combine both feature channels.

In order to compare the uniform-bin pyramid match kernel against the vocabulary-guided

Figure 8-3: Example images from the Caltech-4 database. Six images from each of the four object categories are shown here.

Pyramid matching method	Feature dimension	Mean recognition rate/class	Time/match (s)
Vocabulary-guided bins	128	**99.0**	6.1e-4
Uniform bins	128	64.9	1.5e-3
Vocabulary-guided bins	10	**97.7**	6.2e-4
Uniform bins	10	96.5	5.7e-4

Table 8.1: Caltech-4 object recognition results with uniform or vocabulary-guided bins and the pyramid match kernel.

pyramid match kernel with different feature dimensions, I also generated lower-dimensional ($d = 10$) features using PCA. To form a Mercer kernel, the weights were set according to each bin diameter A_{ij}: $w_{ij} = e^{-A_{ij}/\sigma}$, where i is the pyramid level and j is the bin index (see Eqn. 4.3 (a)). The parameter σ was set automatically as the mean distance between a sample of features from the training set.

Table 8.1 shows the results for this data set. Particularly for high-dimensional ($d = 128$) features, the vocabulary-guided pyramid match is more accurate than the uniform bin pyramid match for these images, and requires minor additional computation. Our near-perfect performance on this data set is comparable to that reached by several others in the literature; the real significance of the result is that it distinguishes what can be achieved with a vocabulary-guided pyramid embedding as opposed to the uniform histograms. Admittedly, here the PCA features yield fairly close performance (96.5% versus 99.0%), which suggests that the dimensionality reduction did not cause much loss in descriptive power. However, we can expect that in cases where high-dimensional features are not well-captured by some low-dimensional manifold, the vocabulary-guided pyramids will be necessary. Finally, on the same data the optimal matching requires $0.31s$ per match, over $500x$ the cost of the pyramid match kernel.

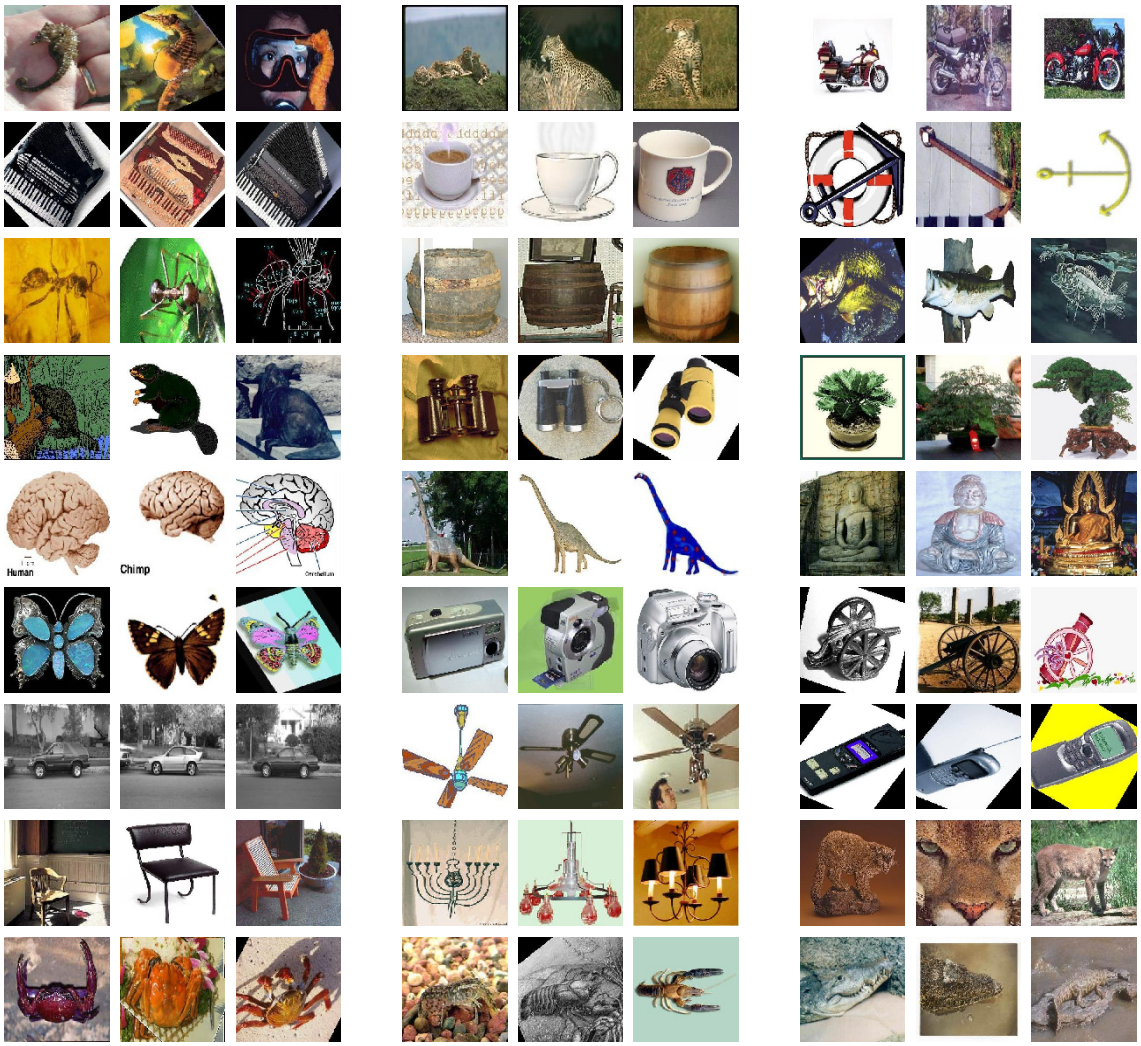


Figure 8-4: Example images from the Caltech-101 database. Three images are shown for each of 27 of the 101 categories.

8.2.3 Caltech-101 Data Set

I also tested our method with a challenging database of 101 object categories developed at Caltech, called the Caltech-101 [31]. The creators of this database obtained the images using Google Image Search, and many of the images contain a significant amount of intra-class appearance variation (see Figure 8-4). In terms of the number of categories, the Caltech-101 is currently the largest benchmark recognition data set available, and as such it has been the subject of many recent recognition experiments in the field. There are 8677 images in the data set, with between 31 to 800 images for each of the 101 categories.

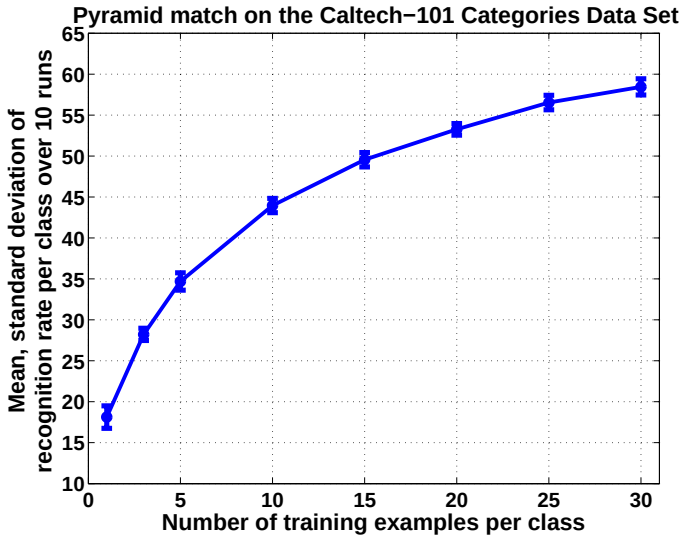
For this data set, the pyramid match operated on sets of SIFT features projected to 10 dimensions using PCA, with each appearance descriptor concatenated with its corresponding positional feature (image position normalized by image dimensions). The features were extracted on a uniform grid at every 8 pixels in the images, i.e., no interest operator was

applied. The regions extracted for the SIFT descriptor were about 16 pixels in diameter, and each set had on average 1140 features. I trained the algorithm using unsegmented images. Since the pyramid match seeks a strong correspondence with some subset of the images' features, it explicitly accounts for unsegmented, cluttered data. Classification was again done with a one-vs-all SVM, and I set the number of grid shifts $T = 3$, and the finest sides of the pyramid to $f = [5, 7, 9]$. I used the current version of the database, which does not contain duplicated images. Note that some images were rotated by the creators of the database, causing some triangular artifacts in the corners of some images; this is how the images are provided to any user.

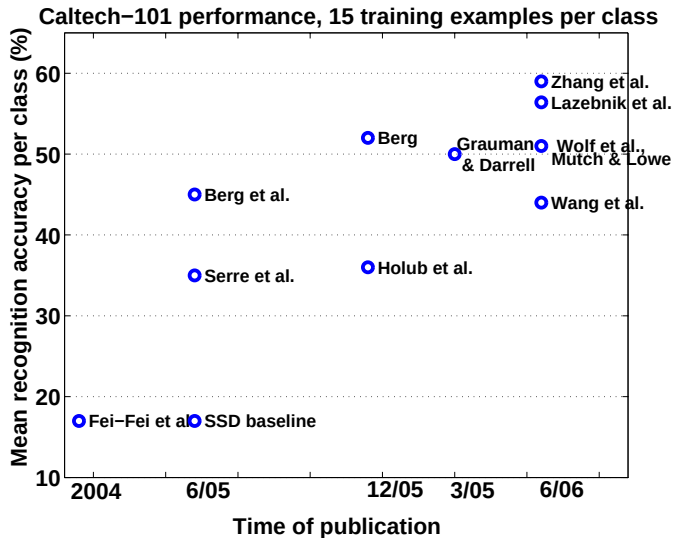
Figure 8-5 (a) shows the multi-class category recognition results using the pyramid match kernel. The recognition scores are given for varying numbers of training set sizes, ranging from one to 30 examples per class. For each training set size, the mean and standard deviation of the pyramid match accuracy over 10 runs is shown, where for each run we randomly select the training examples and use all the remaining database images as test examples. However, all recognition rates have been normalized according to the number of novel test images per class; that is, the mean recognition rate per class is the average normalized score for all 101 categories. Using only one training example per class, the pyramid match achieves an average recognition rate per class of 18%. Note that chance performance with any number of training examples would be just 1%. Using 15 examples (a standard point of comparison), the pyramid match achieves an average recognition rate per class of 50%.

Figure 8-6 shows some of pairs of categories that caused the most confusion for the pyramid match. The examples shown are images that were actually mislabeled by the PMK classifier as belonging to the other category in the pair. Some of these confused category pairs (lotus and water lily, schooner and ketch) have rather subtle distinctions in appearance. Based on confusions caused between the animals here, the gerenuk and kangaroo, we can infer that the local feature description has failed to capture what is important about the global shapes of the two objects; while both creatures share similar-looking torsos, the long legs of the gerenuk appear to not be captured well by the chosen representation.

Figure 8-5 (b) shows all published results on this data set as a function of time since the data was released, including the PMK and results published more recently by other authors [46, 54, 94, 114, 109, 9, 8, 31, 83, 68, 119]. From this comparison we can see that even with its extreme computational efficiency, the pyramid match kernel achieves results that are competitive with the state-of-the-art. We have previously obtained 50% accuracy on average (0.9% standard deviation) when using the standard 15 training examples per category [46]. In addition, Lazebnik and colleagues have also shown that using the pyramid match kernel with sets of spatial features yields very good accuracy on this data set: 56.4% for 15 training examples per class [68]. These results are based on a special case of the pyramid match, where multiple pyramid match kernels computed on spatial features are summed over a number of quantized appearance feature channels. This is in fact the second-to-best performance reported to-date on the data set, and less than three percentage points away from the most accurate result of 59% from Zhang et al. [119], who use a classifier combining support vector machines with nearest-neighbor search and the local geometric blur feature of [10].



(a) PMK categorization results



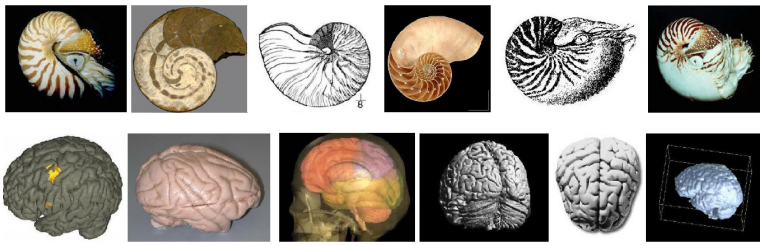
(b) Comparison of all published results

Figure 8-5: Recognition results on the Caltech-101 data set. Plot (a) shows the mean and standard deviation for the recognition accuracy using the pyramid match kernel when given varying numbers of training examples per class, over 10 runs with randomly selected training examples. These are recognition rates that have been normalized according to the number of test examples per class. Plot (b) shows all published results on the same Caltech-101 data set over time since the database's release. Each recognition rate on this plot represents the mean accuracy per class when using 15 training examples per class. Grauman & Darrell refers to our PMK result using appearance descriptors, and Lazebnik et al. refers to a variant of the PMK applied to spatial features in each channel of quantized appearance.

(a) Schooner and ketch

(b) Lotus and water lily

(c) Gerenuk and kangaroo



(d) Nautilus and brain

Figure 8-6: Confused examples from the most confused pairs of categories in the Caltech-101 using the PMK. For each pair of categories shown, these images are examples that the pyramid match classifier misclassified as belonging to the other class. The two category names in each group refer to the top row and bottom row of images for that group, respectively.

Unfortunately most authors for the methods shown in Figure 8-5 do not report the runtimes required, so we cannot directly compare all the varied computational costs alongside these reported accuracy numbers. However, to give some concrete complexity comparisons, we do know that the correspondence-based nearest-neighbor classifier given by Berg and colleagues requires about 5 seconds per match [9], while the variant in [8] requires about 0.04 seconds per match in practice. On the same data, the pyramid match kernel requires 0.002 seconds per match. The method of Berg et al. [9] measures similarity between sets of geometric blur descriptors by approximating the optimal low-distortion correspondences via linear programming, and then uses nearest-neighbors to classify images. It requires $O(m^2 n \log n)$ time to compare two images for n test features and m model features. In [8], nearest neighbor classification is also performed, but on the basis of independent feature voting. Processing one image with voting requires $O(Nm^2)$ time in order to compare its features against all of the features in the N training examples. This breaks down into $O(m^2)$ computation time for every pair of images.

In comparison, pyramid match comparisons require only $O(mL)$ time for the result of 50% accuracy in [46]. For the pyramid match-based result of 56.4% accuracy by Lazebnik et al. in [68] there is an additional one-time $O(mk)$ cost of assigning each of a set's features to one of the k pre-established quantized appearance features; however, once a point set has been mapped to its prototypes, they need not be re-computed to match against additional sets.

The correspondences between local features provide a useful measure of similarity for recognition on this data. But what loss in recognition accuracy do we incur by using the pyramid match approximation instead of the optimal least-cost matching? In Chapter 3 I provided theoretical bounds on the pyramid match's approximation error, and in Chapter 6 I evaluated the approximation error empirically in terms of raw cost differences, ordinal rankings, and relevant rankings. In the following experiment, I evaluate the loss induced by the pyramid match relative to the optimal matching in terms of the generalization accuracy of the classifiers based on either measure.

Given the high values of m that result from the dense feature sampling (on average $m = 1140$ features), it is impractical to use the optimal matching to compute all pairwise kernel values for all 8677 Caltech-101 images for multi-class recognition, as I have done for the pyramid match. However, to make some comparison, I randomly selected 10 pairs of categories, and 18 images from each. For every category pair, I computed the (36×36) pairwise kernel values using both the pyramid match and the optimal matching. As defined in Eqn. 3.5 in Chapter 3, the optimal similarity function computes the least-cost assignment between features in the two sets, then uses a weight for each edge that is inversely proportional to its length. Once the PMK and optimal matching kernel values were computed for each pair of categories, I trained binary SVM classifiers with varying numbers of (randomly selected) training examples per class, with 18 random selections of training sets per training set size. Every SVM instance was then tested using the remainder of the example images as novel inputs.

Figure 8-7 displays 10 plots corresponding to these 10 category pairs; in each the recognition accuracy for the optimal match and the pyramid match are plotted for increasing numbers of training examples. The left side of Figure 8-8 (b) summarizes these plots with a histogram

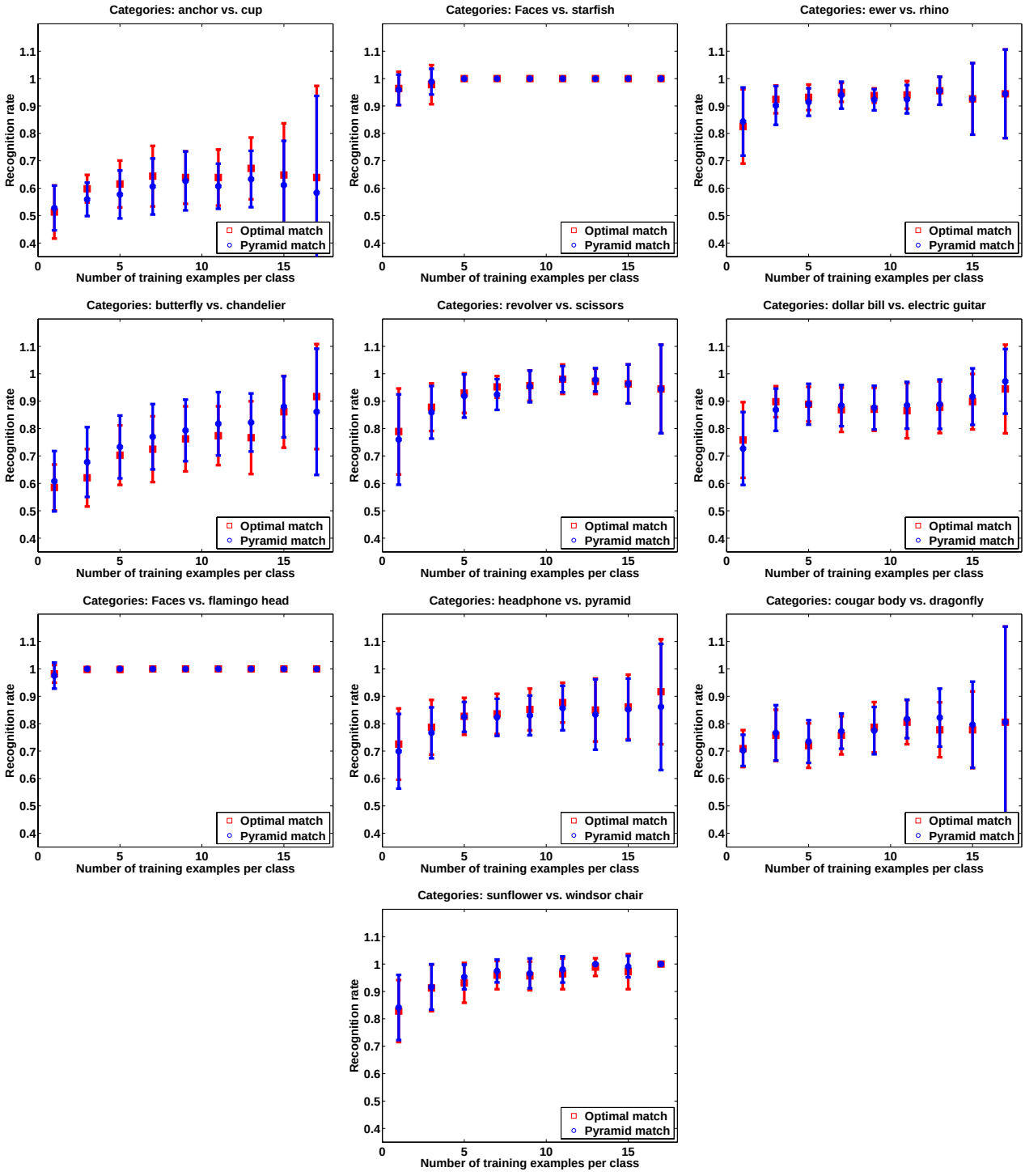
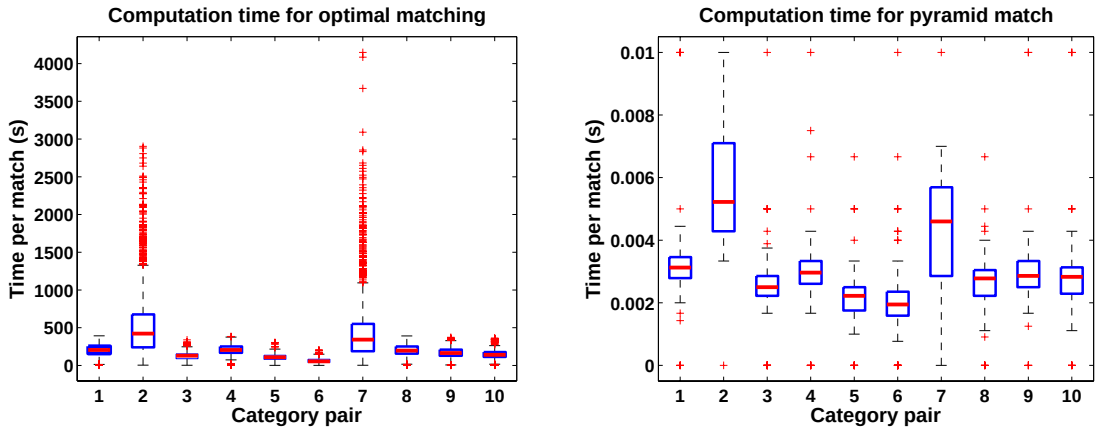
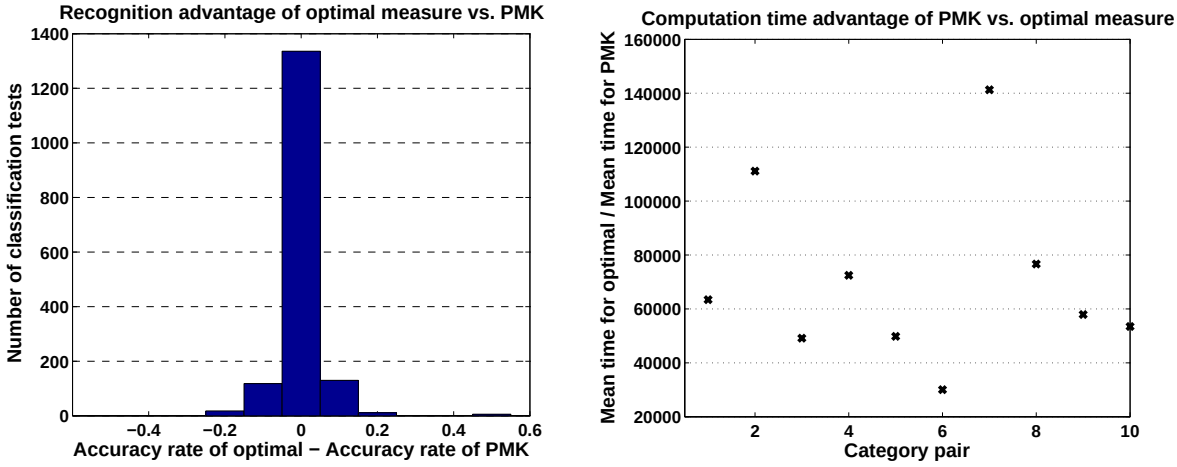


Figure 8-7: Recognition performance on two-class problems from the Caltech-101 data set using the optimal matching and the pyramid match kernel. Points and error bars denote mean and standard deviation of the recognition rate over 18 runs with randomly selected training examples. In each plot, the optimal matching result is shown in red, and the PMK result is shown in blue.



(a) Computation time for optimal match and pyramid match



(b) Tradeoffs for using optimal match or pyramid match on pairs of categories from Caltech-101

Figure 8-8: Part (a) shows the computation time distributions for each category pair, for the optimal (left) and pyramid match measures (right). Note the different vertical axis scales on these two plots. The histogram on the left in part (b) shows the distribution of differences in recognition rates for all instances of the two-class recognition accuracy plots summarized in the previous figure. The right plot in part (b) shows the ratio of the mean time required by the optimal matching to the mean time required by the PMK, for each of the 10 category pairs. In most cases, the optimal measure yields no recognition improvement over the PMK (see the large count over 0 in left plot of (b)), while the pyramid match is several orders of magnitude faster to compute (right plot of (b)).

of the differences between the recognition accuracy of the optimal matching and the PMK over all of the test runs. Both figures reveal that in most cases, the pyramid match accuracy matches that of the optimal measure. In 70% of the cases (1127 of 1620 trials), the pyramid match recognition accuracy is equivalent to that of the optimal match. Overall, the average recognition improvement of the optimal match over the pyramid match is 0.13%, meaning the PMK lost us much less than 1% in the output accuracy score for these categories. Among *only* those cases where the optimal match outperforms the PMK, the improvement in recognition accuracy averages 7.5%.

In addition, categories in this experiment that were "easy" to distinguish with the optimal matching are also recognized consistently with the pyramid match (e.g., the PMK shares the optimal measure's high recognition rates and small variance across runs for the sunflower versus windsor chair, and the face versus flamingo or starfish.) Likewise, category pairs that are more difficult for the pyramid match are also difficult for the optimal measure (e.g., the anchor versus cup).

The boxplots in Figure 8-8 (a) show the computation time required per matching on this data for the optimal matching (left) and the pyramid match (right). These times were all measured running C code on 2.4 GHz processors with 2 GB of RAM. For the optimal matching I used the implementation of EMD provided online by Rubner [92] with the maximum number of iterations set to 4000. In both plots the vertical axis denotes seconds, but the axes are on different scales. The optimal matching required on average four minutes to compare two sets, while the PMK (with $T = 3$) required only 0.003 seconds. On average, the PMK is 70,535x faster. The righthand plot in Figure 8-8 (b) summarizes the complexity advantage of the PMK by showing the mean computation time per comparison ratios for each category pair.

The time distributions for both measures are noticeably higher for category pairs 2 and 7; this is due to the fact that the face category was randomly drawn for both of these pairs, and it contains on average 1000 more features per set than the rest of the categories selected here.

8.3 Learning a Function over Sets of Features

In the following experiments I demonstrate the pyramid match kernel applied to two regression problems: time of publication inference from a collection of research papers, and articulated pose estimation from monocular silhouettes. I use the SVR implementation in the LIBSVM library [17]. In these experiments I use an ϵ-insensitive loss function in order to obtain solutions that are defined in terms of a small subset of the training examples, and which provide good generalization ability. For all experiments, the SVR parameters C and ϵ were chosen using cross-validation.

8.3.1 Estimating a Document's Time of Publication

I have applied the pyramid match kernel to learn a function that maps a bag of local latent semantic features extracted from a research paper (written in a specific field) to an estimate of the paper's time of publication. While much work has been done using bag-of-words representations and latent semantic concept models for text classification and information retrieval, we are not aware of previous work showing *bags of local semantic features*, nor demonstrating direct regression on documents to estimate their publication times.

The bag-of-words model is a widely used representation in which each document is represented by a vector giving the frequencies of words that occur in it, and it has been used in kernel methods [99]. The well-known limitation of such a model, however, is that it encodes nothing about the semantic relationships between words. Each word is treated as an isolated entity, ignoring the fact that distinct words may share degrees of semantic equivalency (e.g., synonyms), or may have different connotations in different contexts (e.g., homonymy). Researchers have therefore adopted latent semantic indexing (LSI) [28] to instill semantic cues into the basic bag-of-words representation. LSI projects data onto a subspace determined using singular value decomposition (SVD) to represent document-term frequency vectors in a lower-dimensional space where semantic relations are better preserved. Generally the subspace for LSI is learned from document vectors that give the frequency of occurrence for each given word (e.g., [25]), which means each document is mapped to a point in a "semantic space" where documents with correlated word frequencies are related.

However, requiring a document to map to a single point in this space assumes that inputs have no clutter (e.g., extra words caused from OCR errors, or text inserted from a webpage ad), and that each document can be represented as a linear combination of the document-level concepts recovered by LSI. Instead, we propose treating documents as bags of *word meanings* by learning the subspace for LSI from *term* vectors, which record the frequency with which a word occurs in each given document. Each document is then a bag of local semantic features, and two documents are compared with the partial matching implied by the pyramid match kernel, i.e., in terms of how well (some subset) of the LSI term-space projections can be put into correspondence. Note that standard kernels (e.g., linear, RBF, polynomial) cannot be used with the bag of word meanings representation, since each word is represented by a real-valued vector. Additionally, the PMK makes it possible to learn a latent semantic space from narrower contexts than entire documents (e.g., paragraphs or sentences) and then represent documents by their component features in this space.

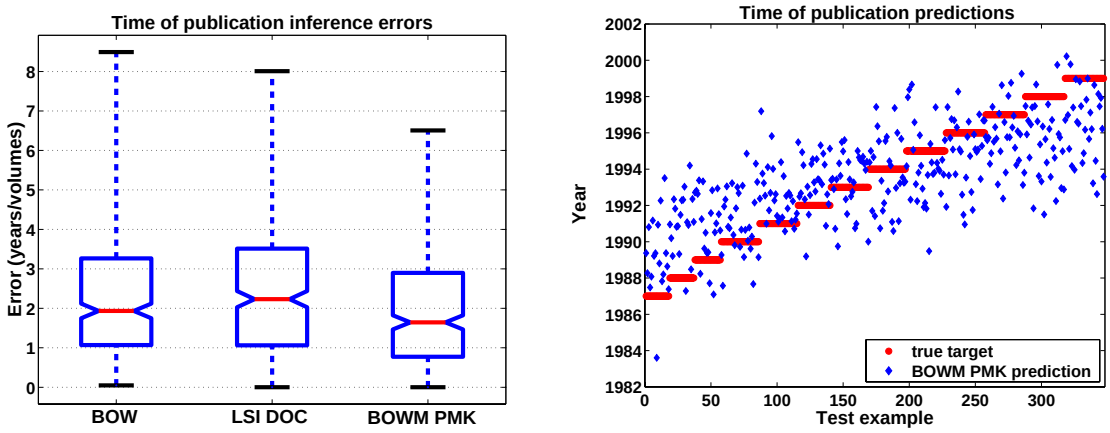


Figure 8-9: Inferring the time of publication for papers from 13 volumes of NIPS proceedings. Boxplots (left) compare errors produced by three methods with a Support Vector Regressor: bag-of-words (BOW) and latent semantic document-vectors (LSI DOC) with linear kernels, and "bag of word meanings" with the pyramid match kernel (BOWM PMK). Lines in center of boxes denote median value, top and bottom of boxes denote upper and lower quartile values, dashed lines show the extent of the rest of the errors. The plot on the right shows the true targets and corresponding predictions made by the pyramid match method (BOWM PMK).

I have experimented with a database containing 13 volumes of Neural Information Processing Systems (NIPS) proceedings—a total of 1,740 documents, available online [84]. For each paper, I extracted the text from the abstract up to (but not including) the references section. While authors' names and bibliography information would likely be good indicators of publication time, they were excluded from the bags of features because we want our method to learn a function indicating topic trends over time, as opposed to a look-up table of dates and names. I applied standard steps to pre-process the text data. Suffixes were removed from words using the Porter stemmer [87], and the WordNet [116] set of stop-list words were removed. Finally, co-occurrence matrices were weighted using term frequency-inverse document frequency (*tf-idf*) to normalize for text lengths and distill the most informative words.

Figure 8-9 shows results for regressing directly on the year of publication for a NIPS paper using the classic bag-of-words approach (BOW), a standard approach applying LSI at the document level (LSI DOC) [25], and the PMK with bags of word meanings (BOWM PMK). All methods were trained with the same randomly selected subset of the data (1393 examples) and tested with the remaining 347 examples. The pyramid match kernel with bags of word meanings performs the best, with a median error of 1.6 years. The PMK values took on average 0.002 seconds to compute. Using a paired difference T-test with $\alpha = 0.01$, I found the difference in performance between our approach and the two existing methods to be statistically significant.

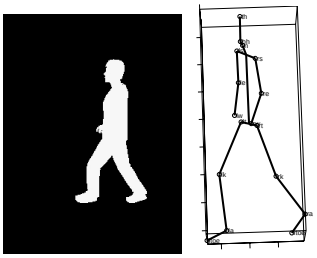


Figure 8-10: A training example generated with graphics software is composed of a silhouette and its corresponding 3-D pose, as represented by the 3-D positions of 15 joints.

8.3.2 Inferring 3-D Pose from Shape Features

In this set of experiments, I use regression with the PMK to directly learn the mapping between monocular silhouette images of humans walking and the corresponding articulated 3-D body poses. Many vision researchers have addressed the difficult problem of articulated pose estimation; recent approaches have attempted to directly learn the relationship between images and 3-D pose [1, 97, 47]. Like these techniques, we learn a function that maps observable image features to 3-D poses.

However, whereas ordered, fixed-length feature sets are required by [47] and [97] (i.e., points are extracted in sequence around the contour, or features are taken from fixed image windows), my method accepts unordered features and inputs that may vary in size. This is a critical difference: images will naturally have varying numbers of features (due to occlusions, clutter, translations, shape deformations, viewpoint changes, etc.), and a robust global ordering among features within a single view may not be possible in the presence of viewpoint and pose variations.

In the pose estimation method of Agarwal and Triggs [1], local features are mapped to pre-established prototypes, and every image is represented by a frequency vector counting the number of times each prototype occurred. A relevance vector machine is then trained using these vectors with a Gaussian kernel. While this type of approach allows unordered features, it can be sensitive to clutter, as I will show below.

Note that in the shape retrieval experiments of Chapter 7, the human silhouette images contained people in a wide variety of arbitrary poses, and the objective was to retrieve examples having similar overall shape. In these experiments, the space of poses is more constrained—only images of people walking are used—and the objective is to use the similarity measured between shapes to learn the *function* from silhouettes to pose.

For a training set, I used the graphics package Poser [21] to generate 3,040 images of realistic synthetic images of pedestrians. Each image was rendered from a humanoid model with randomly adjusted anatomical shape parameters walking in a randomly selected direction. For each image, both the silhouette and the 3-D locations of 15 landmarks on the model's skeleton corresponding to selected anatomical joints were recorded (see Figure 8-10). Regressors are trained with silhouette inputs and produce 3-D joint position outputs. Once regressors have been trained with the synthetic data, we can use them to perform regression with either additional synthetic examples (for which we have ground truth poses), or with

real image data (for which we can only evaluate predicted poses qualitatively).

Each silhouette is represented with a set of local contour descriptors. At each contour point, we extract a shape context histogram [7], which bins nearby edge pixels into a log-polar histogram, thereby encoding local shape. For each shape context histogram, I used 12 angular and five radial bins with a fixed scale to capture only local points relative to each edge point. To form a more compact descriptor, I used 5-D PCA projections of the initial 60-D shape context histogram features. Thus each silhouette shape is represented by an unordered set of shape context subspace features, and each set varies in size due to the varying number of contour points per image. Note that although this representation does not contain explicit spatial constraints, the overlap between neighboring shape context histograms provides an implicit encoding of how the features should be related spatially.

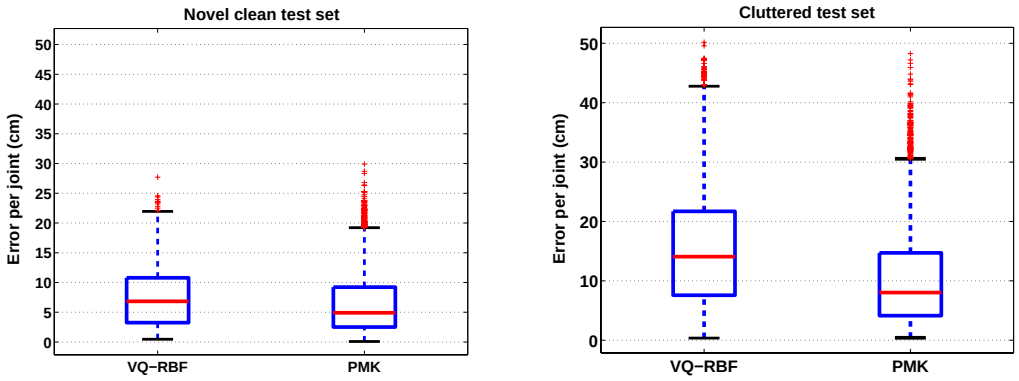
The number of contour points (and thus features) per image varied from 405 to 878. The multi-resolution histograms used by the pyramid match contained ten resolution levels, as determined by the diameter of the features in the training examples, and each contained on average 2644 non-empty bins. Computing a pyramid match kernel value required about 0.002 seconds. For each dimension of the pose targets, I trained an ϵ-insensitive SVR using the pyramid match kernel matrix between the training examples. Each SVR had on average 308 support vectors.

As a baseline, I also implemented a method that uses frequency vectors over feature prototypes to represent each image, using [1] as a guideline. Vector quantization (VQ) is performed on the shape context subspace features found in the training set to establish a set of 100 prototype features. Then all of the features detected in a new image are mapped to a 1-D frequency histogram over these prototypes using soft voting with Gaussian weights. A Gaussian RBF kernel is then applied, with γ chosen based on the maximum inter-feature distance. In the following I will refer to this baseline as VQ-RBF.

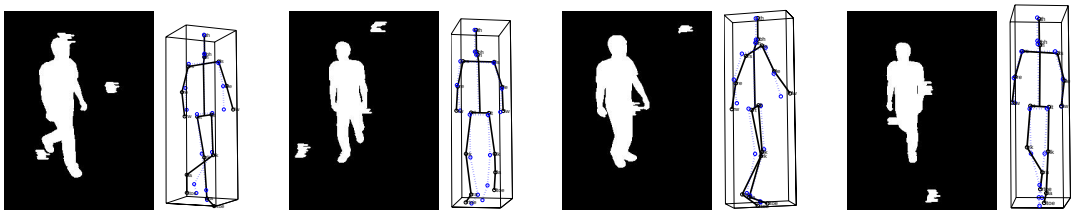
For a novel test set of 300 Poser-generated silhouettes, the pose inferred using the PMK had a median error of 4.9 cm per joint position. For the same test set, VQ-RBF obtained a slightly worse median error of 6.8 cm (see Figure 8-11). Using a paired difference T-test with $\alpha = 0.001$, I found the difference in performance between the two methods to be statistically significant.

The silhouette contours produced with Poser are of course very "clean", that is, the shapes are perfectly segmented since they were formed by directly projecting the CG body. While it is reasonable to train a model with this well-segmented data, we can expect real-world examples to exhibit poorer foreground-background segmentations due to occlusions, clutter, shadows, or backgrounds that look similar to the foreground object. Therefore, I altered the silhouettes for a separate test set of 300 examples to introduce clutter and occlusions; starting with a Poser silhouette, I generated one to eight random blob regions in the image for which the foreground/background labels were swapped. The blobs' positions, sizes, and shapes were generated randomly. The result is a test set that mimics real-world occlusions and clutter, yielding imperfect contours for which estimating pose is more challenging (see Figure 8-11).

On the cluttered test set, our method inferred poses with a median error of 8.0 cm per joint, while VQ-RBF had a median error of 14.1 cm (see Figure 8-11). Again, using a



(a) Pose estimation errors



(b) Cluttered synthetic test examples

Figure 8-11: Pose inference results. The top row (a) gives a quantitative evaluation of performance on synthetic data with ground truth. The boxplots compare the error distributions for the pyramid match kernel (PMK) and an RBF kernel over prototype frequency vectors (VQ-RBF). Errors are measured by the distance between the 15 true and inferred joint positions in each image. Results for two test sets are shown: a set of novel, clean silhouettes (left plot), and a set with randomly generated clutter or extra foreground blobs (right plot). The bottom row (b) shows example poses inferred by our method from synthetic cluttered silhouettes. In each, the true pose (solid black) is overlayed with the estimate (dotted blue). These examples contain average case errors.

paired difference T-test, I found the difference in performance to be statistically significant: with 99.99% confidence, the pyramid match yields average errors that are smaller than those of VQ-RBF by amounts between 4.5 and 5.2 cm per joint.

This experiment demonstrates the pyramid match kernel's robustness to superfluous features in an input. The blobs added to the cluttered test set introduced extra contour features to the silhouettes, and they altered parts of the true contour in cases where they overlapped with the real silhouette. The VQ-RBF distance over prototype frequency vectors essentially penalizes any features that have no "match" in a support vector training example's set of features. In contrast, the pyramid match's partial matching rewards similarity only for the features placed into correspondence, and ignores the extraneous clutter features. This is an important advantage for many vision applications, where segmentation errors, viewpoint changes, or image noise will often affect which features (and how many) are detected.

Finally, I applied our method to a test set of real images of various subjects walking through a scene. A basic background subtraction method was used, which resulted in rough segmentations; body parts are frequently truncated in the silhouettes where the background is not highly textured, or else parts are inaccurately distended due to common segmentation problems from shadows. Ground truth poses are not available for this test set, but Figure 8-12 shows some example output poses inferred by our method.

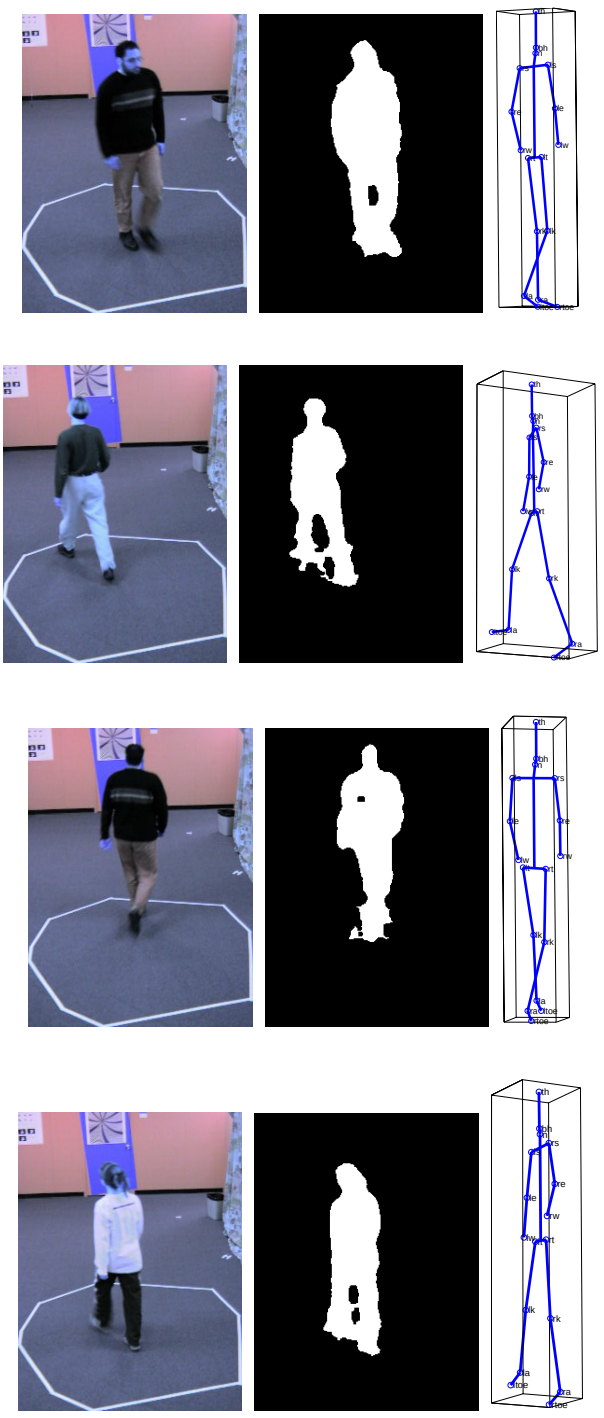


Figure 8-12: Pose inference on real images of people walking. A function to estimate body pose is learned with the pyramid match kernel from synthetically generated examples. Then given novel real images (left column), a simple background subtraction is used to extract the silhouette shapes (middle column). Local contour features are extracted from these silhouettes. Given a set of these unordered contour features, the learned function returns a list of 15 3-D joint positions (right column).

Chapter 9

Unsupervised Category Learning from Sets of Partially Matching Features

Current approaches to object and scene recognition typically require some amount of supervision, whether it is in the form of class labels for training examples, foreground-background segmentations, or even a detailed labeling of objects' component parts. In part due to the significant expense of providing these manual annotations, such approaches are in practice restricted to relatively small numbers of classes and/or few training examples per class. Additionally, human supervision may result in unintentional biases that can be detrimental to generalization performance. An unsupervised (or semi-supervised) technique that is able to recover salient categories directly from images would relieve these burdens and possibly offer new insights into image representation choices.

In this chapter I present a method to automatically learn object categories from unlabeled images. Each image is represented by an unordered set of local features, and all sets are embedded with the pyramid match into a space where they cluster according to their partial-match feature correspondences. After efficiently computing the pairwise affinities between the input images in this space, a spectral clustering technique is used to recover the primary groupings among the images.

I introduce an efficient means of refining these groupings according to intra-cluster statistics over the subsets of features selected by the partial matches between the images, and based on an optional, variable amount of user supervision. The consistent subsets of feature correspondences within a grouping are computed to infer category feature masks. The output of the algorithm is a partition of the data into a set of learned categories, and a set of classifiers trained from these ranked partitions that can recognize the categories in novel images.

Implied in the motivation for unsupervised learning of categories is the idea that while labeled data is expensive and must be used frugally, unlabeled data is generally inexpensive to obtain in large quantities. Thus a critical criterion for a method intended to learn from

large amounts of unlabeled data is computational efficiency; by leveraging the PMK in this context, I have designed a method that will scale well with both the amount of input data as well as the size of the inputs themselves.

Possible applications of the proposed method include learning object class models from unlabeled data, shot matching or scene grouping from video sequences, and content-based refinement of keyword-based image retrievals. In this chapter I demonstrate the applicability to learning object categories to allow unsupervised training of discriminative classifiers.

9.1 Related Work on Unsupervised Category Learning

Existing approaches to this problem use vector quantization to build a codebook of feature descriptors, and then transform each set input to a single vector counting the number of occurrences of each prototype feature. Conventional clustering methods or latent semantic analysis (LSA) may then be directly applied [89, 101, 33], and have been shown to yield promising results when learning object or scene categories or filtering keyword-based image retrieval outputs.

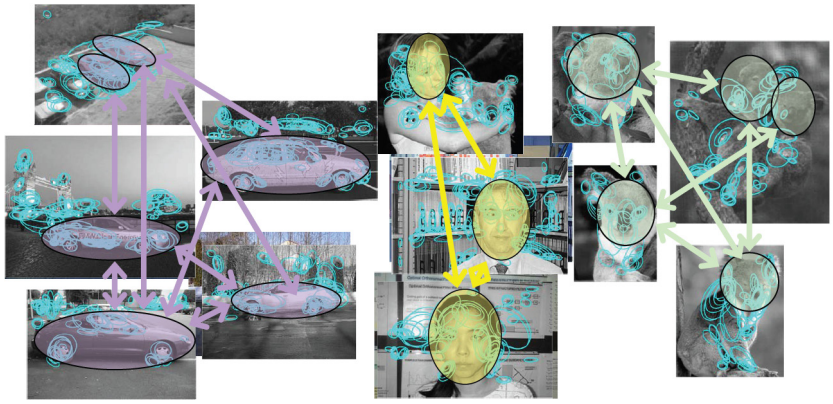
However, such approaches do not explicitly allow for "clutter" features caused by image backgrounds or occlusions, and the need to pre-compute a codebook raises computational complexity and data availability issues. In addition, it is not clear how existing techniques could accommodate the addition of small amounts of labeled data or *a priori* knowledge about pairwise constraints between particular unlabeled examples.

As discussed earlier, in general methods to solve for explicit correspondence are computationally expensive, requiring cubic time to form globally optimal assignments, and even more for assignments including higher order constraints between features. Some approximate methods have been defined, which offer improved performance under certain restrictions. Berg et al. introduced a powerful recognition algorithm that uses linear programming to solve for approximate correspondences, and they showed how to use the correspondence-based metric to find regions of common spatial support for objects in labeled training examples, thus avoiding the need for manually segmented images [9]. In Chapter 8 (and [44]), I showed the strength of the pyramid match kernel for discriminative classification when inputs are variable-sized sets of features. However, in both recognition frameworks, it was assumed that class labels were provided for all training images.

9.2 Grouping Feature Sets with Partial Correspondences

Every input image is decomposed into some number of local appearance features, where each feature is a vector descriptor for the local region or patch. So given an unlabeled data set $\mathbf{U} = \{I_1, \ldots, I_N\}$ containing N images, image I_i is as before represented by a set $\mathbf{X}_i = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_i}\}$, where $\mathbf{x}_j$ is a descriptor vector, and m_i may vary across images in $\mathbf{U}$ depending on the number of features detected in each image. In my implementation I chose to use Lowe's SIFT descriptor [73], but other options are certainly possible.

(a) Pyramid match graph



(b) Graph partition

Figure 9-1: Clustering according to partial match correspondences.

The initial image groupings are formed by embedding the feature sets into a space where they cluster according to their partial-match correspondences (see Figure 9-1). I use the pyramid match kernel to efficiently obtain these matchings. Comparing sets of image descriptors in this way provides an efficient (linear in the number of features m_i) measure of how well the two sets' features may be put into correspondence. That the matching formed is partial is useful when we want to learn from unlabeled images containing multiple classes, varying backgrounds, and occlusions—cases where portions of the feature sets may be considered outliers that should not affect the matching quality.

The pairwise pyramid match affinities over feature sets serve to form an undirected, fully-connected graph over $\mathbf{U}$: nodes are images, and edges are weighted according to partial-match similarity between the images' feature sets. Within this graph, we would like to discover categories from those images with the strongest aggregate feature matchings. We seek the partitioning of the nodes that will preserve strongly connected groups while dividing nodes with minimal joint correspondences.

To this end, we employ spectral clustering and use the normalized cuts criterion developed by Shi and Malik for image segmentation [100]. The algorithm "cuts" the nodes into disjoint sets by removing connecting edges; the optimal partitioning both minimizes the amount of dissociation between groups and maximizes the association within groups. The normalized cut dissociation measure is essentially designed to remove edges between the least similar nodes without favoring cuts that partition out small numbers of isolated nodes. In this case, that means prohibiting a few images that happen to have exceptional feature matchings from being selected as categories when there exist broader range associations between feature sets.

Though minimizing the normalized cut is NP-complete, the authors provide an efficient approximate solution based on solving an eigenvalue problem,

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{K})\mathbf{D}^{-\frac{1}{2}}x = \lambda x, \tag{9.1}$$

where $\mathbf{K}$ is an $N \times N$ affinity matrix over data nodes $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$, $\mathbf{D}$ is an $N \times N$ diagonal matrix containing the sums of the rows of $\mathbf{K}$, and x is an N-dimensional indicator vector used to obtain the bi-partition of the graph. To form multiple partitions, recursive cuts or multiple top eigenvectors are used. Extracting the normalized cuts grouping over the pyramid match affinity matrix $\mathbf{K}$ with entries $K_{ij} = \mathcal{P}_\Delta(\mathbf{X}_i, \mathbf{X}_j)$ for all images in $\mathbf{U}$ thus provides our initial set of learned categories.

This framework allows the introduction of weak semi-supervision in the form of pairwise constraints between the unlabeled images. Specifically, a user may specify "cannot-group" or "must-group" connections between any number of pairs in the data set. Following the paradigm suggested in [62], we modify the graph over $\mathbf{U}$ to incorporate this information to assist category learning: entries in the affinity matrix $\mathbf{K}$ are set to the maximal (diagonal) value for pairs that ought to be reinforced in the groupings, or set to zero for pairs that ought to be divided.

For data sets with a large number of example sets to be clustered, we can avoid computing all $O(N^2)$ affinities and obtain a more efficient estimate of the pyramid match kernel matrix by employing the Nyström approximation technique [113].

9.3 Inferring Category Feature Masks

Due to the nature of a partial matching, the clusters produced with normalized cuts risk containing non-homogenous members. While ignoring superfluous features without penalty to the matching similarity is desirable in the sense that it allows a degree of tolerance to clutter, outlier features, and noise, it also means that sets containing similar backgrounds may be allowed to match just as well as those containing similar objects of interest. Likewise, images containing multiple objects may find strong partial matchings with examples containing single objects from each class, thereby confounding the normalized cuts criterion in some cases (see Figure 9-2).

To address this, we look to the pattern of correspondences within each cluster, and leverage the information contained in the intra-cluster partial matching statistics to refine the initial

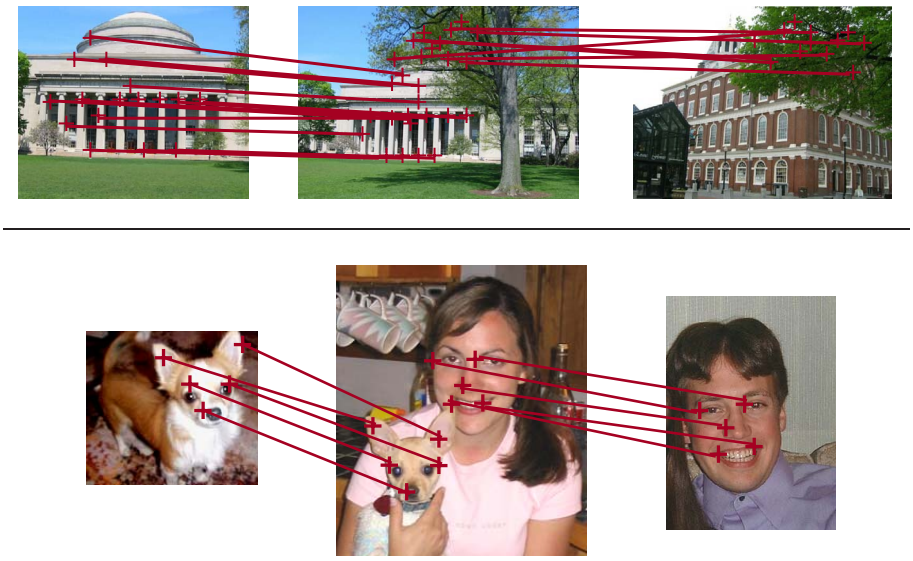


Figure 9-2: Graph partitioning according to partial matchings may allow problematic groups, for example when background features and foreground features find good matchings in different categories of images. In the top row, the image-to-image similarity between the right and center images may be indistinguishable from that of the center and left images, even though the right image is matching what are background features for the domed building category. In the bottom row, the presence of two categories in the center image causes it to match equally well to the images on its left and right, which contain individual instances of those categories. As a result, graph partitioning algorithms may be unable to make appropriate cuts.

grouping. The goal is to identify prototypical cluster members (or, conversely, outlier cluster members) by computing for each example the distribution of its features that were used most heavily to form matchings with other examples within the cluster. The intuition is that we expect "inlier" images to utilize similar portions of their feature sets to form partial matches with one another, while outlier cluster members will cause most images within the cluster to contribute an inconsistent subset of features relative to their other matchings.

To apply this concept, we require the inter-feature correspondences for the pairwise partial matches within each cluster. We can use the implicit correspondences of the pyramid match to induce explicit approximate correspondences between two images, as described in Section 3.2.3. The pyramid match considers feature points matched at the finest resolution pyramid level in which they share the same bin. This means that in any bin where two point sets both contribute points, the points from the set with fewer points in that bin are certainly matched, but only some (unknown) subset of points is matched from the set having more points in that bin. If the counts are equal in a given bin, all points falling in that bin from both sets are matched to each other, in some permutation.

When computing the multi-resolution histograms for an input set of descriptors, we attach to each bin index the indices of the features that the bin spans. This allows us to trace feature matchings during the summation over histogram intersections in Eqn. 3.10 based on which specific points are responsible for causing an intersection at a particular level.

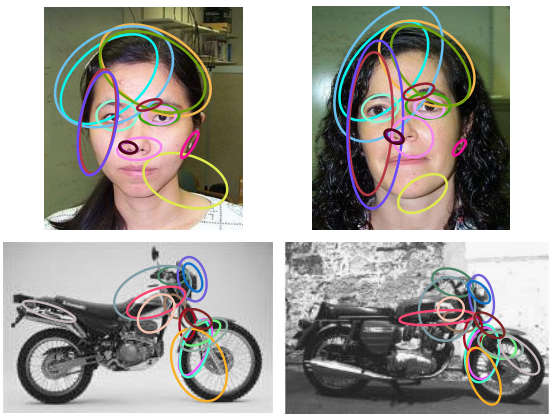


Figure 9-3: Examples of explicit feature correspondences extracted from a pyramid matching. Displayed here are the most confident matches found for two image pairs, as denoted by the color-coded elliptical feature regions. (This figure is best viewed in color.)

For each input $\mathbf{X}_i$, a weighted indicator vector $\mathbf{r}_i$ of dimension m_i is maintained, i.e., $\mathbf{r}_i$ is indexed by the input's feature set. Each indicator is initialized to zero, and then at each level of the pyramid intersections it is updated to reflect the new matches formed.

The partitioning of the feature space provided by the pyramid decomposes the required matching computation into a hierarchy of smaller matchings. Upon encountering a bin with a nonzero intersection value, an explicit matching is computed between only those features from the two sets that fall into that particular bin. Given this matching, entries in the two input sets' indicator vectors corresponding to that bin's attached feature indices are recorded as the similarity between the respective matched points. All points that are used in that per-bin optimal matching are then flagged as matched and may not take part in subsequent matchings within larger bins at coarser resolutions of the pyramid. (Results evaluating the quality of the approximate correspondence fields as compared to the globally optimal correspondences were provided in Chapter 6.)

The result is one weighted indicator vector per image, per matching comparison that reveals both which features were used in each partial match, as well as how strongly those features played a role in the total matching cost (see Figures 9-3 and 9-4). We use these indicator vectors as feature masks that designate which component features each set contributed to matchings. For each image in a cluster containing C members, a typical feature mask is computed as the median indicator vector over that image's $C-1$ within-cluster matchings.

9.4 Identifying Prototypes

To refine the groupings provided by normalized cuts clustering, the pyramid match affinities are re-computed between cluster members using the median feature masks to weight the input feature sets. That is, rather than entering unit bin counts for each feature during the pyramid computation, each feature adds a mass to the bin that is proportional to its weighting in the median feature mask for that example. Essentially this re-weights the

Figure 9-4: Inferred feature masks for a face category. The elliptical regions indicate where features were extracted from the images based on the Harris-Affine interest operator. The boundaries of these regions are color-coded in order to show which features contribute most strongly to each image's matchings against the other face images: blue (darker colored) ellipses denote the features in each image with the high weights in the mask, and yellow (light colored) ellipses denote the remaining features, which have low weights in the mask. Each entry in an inferred feature mask reflects how consistently that feature can be matched against all other images in a cluster. These examples demonstrate how the inferred feature masks reveal which parts of the images correspond to the in-class category, and can be used to downplay the impact of background or clutter features in the matching. (This figure is best viewed in color.)

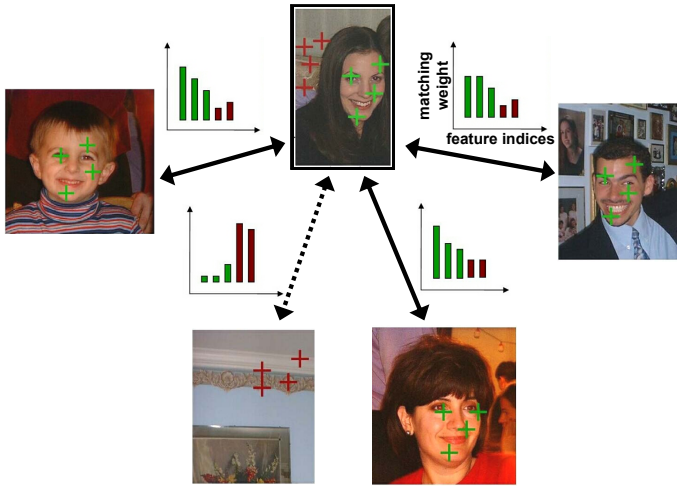


Figure 9-5: A schematic view of category feature mask inference. Within a single cluster, outlier images are detected by considering the typical per-image feature masks implied by which component features of an image contribute most strongly to partial matchings with other members of the cluster. In this illustrative example, the similarity between the matched-feature distributions among the faces reveals the outlier non-face image, whose features happen to match the background of the top image. Shown here are the four matched-feature distributions for the top center image against the rest, with the in-mask features colored green, and non-mask features colored red. Re-weighting the correspondences according to the example's median indicator mask causes the similarity against the outlier image to be downgraded, as indicated by the dashed line. To deduce cluster outliers, feature masks are determined using all pairs in this manner. (This figure is best viewed in color.)

individual feature matchings to favor those that are established between features likely to belong to the "inlier" cluster examples, and to downplay those caused by the inconsistent outlier matchings (see Figure 9-5). This new $C \times C$ affinity matrix is left un-normalized, since given the feature masks we no longer wish to treat small correspondence fields as being equally significant to large ones.

Having adjusted the within-cluster affinities to take the feature masks into account, we can then sort the images in each group according to how consistently they match the remainder of the group. I define the flow per example within a cluster to be the sum of its re-weighted pyramid match scores against the rest of the cluster members. Items in the cluster are then ranked according to their flow magnitudes, and examples falling within a specified top percentile of this ranking are identified as *candidate prototypes*. In my implementation I have evaluated the categories learned with no supervision under various settings of the percentile parameter, but one could also envision allowing minimal semi-supervision at this stage, where a user could be presented with a small number of prototypes to label. Should the user disagree with the cluster labels, we could introduce link constraints into the re-weighted cluster affinity matrices here (as well as prior to performing normalized cuts) and iteratively recompute the prototypes.

The prototypes are then considered the best representatives for the particular learned cat-

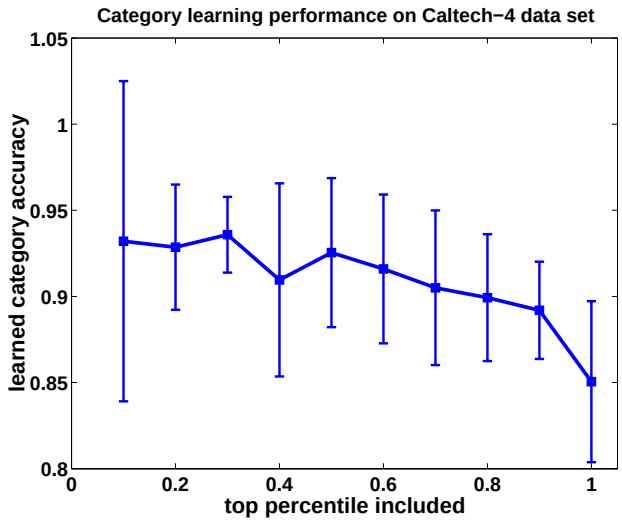


Figure 9-6: Accuracy of categories learned without supervision, as measured by agreement with ground truth labels. The percentiles determine the amount of prototype candidates to keep per learned class, and the results shown here are averaged over 40 runs for each percentile. The plotted points denote the mean performance for those runs and error bars denote the standard deviation. See text for details.

egory, and may be used to build a classifier that can predict categories for novel images. In my implementation I have chosen to use a discriminative kernel-based classifier, the Support Vector Machine, since it can directly use the same pyramid match kernel matrix as the normalized cuts computation; however, other alternatives are equally plausible.

9.5 Category Learning Results

In this section I present results evaluating the proposed method when applied to perform unsupervised or semi-supervised learning of object categories, and I show its ability to automatically train a classifier that can be used to predict the labels of unseen images.

I have experimented with a common benchmark data set containing four object classes, the Caltech-4 database, which is comprised of 1155 rear views of cars, 800 images of airplanes, 435 images of frontal faces, and 798 images of motorcycles. (See Chapter 8 for results on this data set using supervised learning and the PMK.) Many of the images of the airplanes and motorcycles contain white borders which I removed before any processing was done, so as to avoid inserting features that might provide either misleading or helpful cues to our algorithm. I detected salient points in the images with a Harris-Affine interest operator [79], and decomposed images into sets of SIFT features [73], scale-invariant descriptors based on histograms of oriented image gradients. More compact (10-dimensional) features were obtained from the original SIFT descriptors using PCA.

For the first experiment, I provided our method with a pool of unlabeled images containing examples from each class and requested that it learn four categories. Figure 9-6 summarizes the accuracy of the groupings produced as a function of the percentage of prototypes ex-

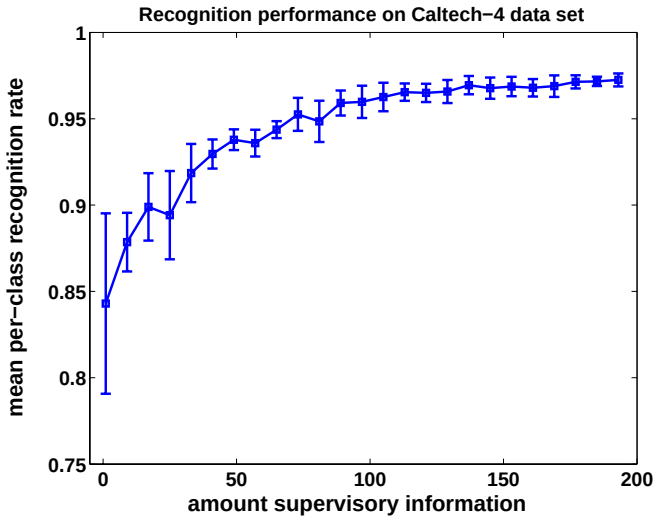


Figure 9-7: Recognition performance on unseen images using categories learned with varying amounts of weak semi-supervision. The horizontal axis denotes the number of (randomly chosen) "must-group" pairings provided, and the vertical axis denotes the recognition performance averaged over four classes. The plotted points are mean values and error bars are the standard deviations over 40 runs with randomly selected training/testing pools. See text for details.

tracted from each initial normalized cuts grouping as discussed above. Accuracy is measured as the mean diagonal of a confusion matrix computed for the learned categories against the ground truth labels. For each percentile level tested, I ran the method 40 times, each time with a different random subset of 400 images from the 3188 total images, with 100 images per class. This result demonstrates the impact of the refinement mechanism to detect prototypical cluster examples based on the inferred feature masks. Figure 9-4 shows some examples of inferred category masks for face images with cluttered backgrounds from the Caltech data set.

I have also evaluated how the categories learned with our method will generalize to predict labels for novel images. I trained Support Vector Machines with the pyramid match kernel using the labels produced with varying amounts of semi-supervision (see Figure 9-7). Recognition performance is measured as the mean diagonal of a confusion matrix computed for a total of 2788 novel test images of the four classes (ranging from about 300 to 1000 test images per class), and results are averaged over 40 runs with different randomly selected pools of 400 unlabeled "training" images. Semi-supervised constraints are of the "must-group" form between pairs of unlabeled examples, and an equal number of such constraints was randomly selected for each class from among the training pool. The results suggest that the category learning stands to gain from even rather small amounts of weak supervision.

Sivic and colleagues have developed an unsupervised approach to category learning [101] that is based on the probabilistic Latent Semantic Analysis (pLSA) model of Hofmann [51, 52]. In their technique, a set of local features is transformed into a single word frequency vector, based on a pre-determined local patch vocabulary found with vector-quantization. Then, probabilistic Latent Semantic Analysis is applied to automatically recover the un-

derlying visual "topics". Sivic et al. obtain 98% clustering accuracy using pLSA on the Caltech-4 database. They also provide a baseline performance of 72% obtained using k-means clustering on the word frequency vectors. With no supervision at all, we obtain a result of 85%, which is about halfway between the k-means and pLSA results. Our semi-supervised approach can match the 98% performance on the entire data set, though this is with about 175 random pairwise constraints provided (see Figure 9-7).

Since we have selected different feature types than the authors of [101], in the future it will be interesting to see what impact alternative feature types might have on the grouping accuracy. While the approach of Sivic et al. requires clustering feature data to initially establish a visual word vocabulary, with the pyramid match we compare feature sets directly and thus do not have this start-up cost. Additionally, alternative classifiers in the back-end of the method presented here may prove to be more effective; as Figure 9-6 shows, without any supervision, the prototypes automatically discovered with our approach are on average 94% correct. It is possible, for example, that training exemplar-based classifiers with these strong prototypes would prove more effective than the discriminative classifier applied in these experiments.

In these experiments, the number of groupings formed by normalized cuts was specified as the number of classes expected in the data set; however automated model selection techniques could be applied (see for instance, [118]) to remove this user-supplied parameter.

Chapter 10

Conclusions

In this dissertation I have introduced computationally efficient techniques that effectively handle comparisons, learning, and indexing for the set-of-features representation. I have demonstrated my approach successfully on a variety of data sets for vision tasks related to content-based image retrieval and object recognition.

A central contribution of this work is the pyramid match algorithm, a linear-time method for computing an implicit partial matching between two unordered sets of vectors. I have shown both theoretically and empirically that the pyramid match approximates the cost of the optimal partial matching. Unlike previous matching approximations that were limited to inputs with equal numbers of features, the pyramid match can accept variable-sized inputs. This is critical for image matching, where the number of detected salient features generally varies per example. Additionally, since the pyramid match allows partial matchings, it tolerates clutter features without any penalty to the matching score.

I have shown how an approximate matching can benefit from knowledge of underlying structure in the feature space. The vocabulary-guided pyramid match exploits the structure when hierarchically partitioning the feature space, resulting in a data-dependent pyramid that can remain accurate even for high-dimensional feature spaces. Whereas previously suggested approximations suffer from a linear decrease in accuracy with the feature dimension, in practice the vocabulary-guided pyramid match is consistently strong for spaces of over 100 dimensions.

In previous work, randomized hashing algorithms for normed spaces [38] and hash function families specialized for particular distances of interest [57, 19] have been developed. I have drawn on existing techniques to demonstrate sub-linear time approximate similarity search over a matching between equally-sized sets of local shape or appearance features.

Furthermore, I have introduced a new embedding for the pyramid match that makes it possible to perform sub-linear time hashing over a matching between variable-sized sets of vectors. While previous techniques would necessitate altering the feature distribution of variable-sized sets in order to fit the constraints of existing hashing methods, the embedding I have proposed allows us to maintain the feature distributions of input sets. The embedding is also by definition robust to outlier feature values. I have proven that locality sensitive

hashing is not possible with a partial match normalized by the minimum input cardinality.

Through several content-based image retrieval experiments, I showed the power of approximate similarity search over sets of local features for querying very large image databases. While correspondences between local parts are widely held to provide a robust reflection of image or shape similarity, the expense of previous techniques prevented them from scaling to realistically sized databases. Earlier results for correspondence-based shape and image matching have only been shown for relatively small data sets. In contrast, in this work I have demonstrated reliable matching on a database exceeding 130,000 examples. The computational savings is dramatic. For example, in cases where exact matching methods would require hours to return a result, my approach returns relevant examples in less than a second. This is an important gain, since this speed and scalability is imperative to drive applications that demand real-time query responses, such as situated search on a mobile device.

Within my content-based image retrieval experiments, I evaluated standard alternative approaches for handling unordered sets of image features. A side-by-side comparison of voting, a bag-of-prototypes technique, and correspondence-based matching revealed that—computational costs aside—particular types of image data may be more suited to each different kind of similarity measure. In a test using images of repeated scenes, voting prevailed due to the distinctive, reliably detected features. On the other hand, the approaches which consider the distribution of features in a set at once (both the bag-of-prototypes and matching approaches) outperformed voting on tasks that seem to require knowledge of the features' co-occurrence: comparing textures and object categories with some appearance variation.

I have shown that the pyramid match forms a Mercer kernel, which makes it suitable for use in kernel-based learning methods, including those that have convergence guarantees requiring a Mercer kernel. This connection is quite powerful, since we can use the pyramid match kernel to access a wide class of existing kernel methods with robust local image feature representations. In Chapter 2 I outlined the conceptual differences between existing set kernels and the pyramid match kernel (PMK); one important distinction is that the PMK requires time linear in the number of features per set to compute, while others require cubic or quadratic time.

The PMK's low cost relative to input size is a significant advantage for recognition. While other methods will in practice be forced to artificially limit the quantity of local descriptors, the PMK offers a scalable solution to correspondence-based learning. In my object categorization experiments I demonstrated how critical rich, dense local representations can be to recognition accuracy. I have shown very strong performance using the PMK to recognize categories in several well-studied data sets, but for a significantly lower computational cost than other state-of-the-art techniques.

I further demonstrated the flexibility of the PMK by showing how we can learn functions over the set-of-features representation for some parameter of interest. In my experiments, regression on two types of data with the PMK—human figure contours and sets of word meanings in text documents—yielded accurate results relative to existing methods. A comparison against a bag-of-prototypes approach showed that the PMK is more tolerant of clutter in the form of foreground-background estimation errors; its partial match compar-

isons allow clutter features to be ignored, whereas bag-of-prototypes comparisons consider an entire feature distribution at once.

Finally, I have proposed an approach to automatically learn object categories from unlabeled images. I leverage the partial matching between sets of local features to perform unsupervised clustering, and I have introduced an efficient means of refining groupings according to intra-cluster statistics over the subsets of features selected by the partial matches between the images. While previous techniques have shown how to select the most discriminative or salient features for a given category when images have class labels [9, 111, 34], my approach makes it possible to infer category feature masks for images that have no labels. In contrast to category discovery methods based on probabilistic Latent Semantic Analysis [89, 101, 33], my approach allows the introduction of variable amounts of semi-supervision.

While I have concentrated on vision-related applications, the techniques I have developed are generally applicable to matching and indexing problems with sets of vectors; my research is completely suitable for applications with alternate image feature types or even in alternate learning domains.

Bibliography

[1] A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2004.

[2] P. Agarwal and K. R. Varadarajan. A Near-Linear Algorithm for Euclidean Bipartite Matching. In *Symposium on Computational Geometry*, 2004.

[3] S. Agarwal, A. Awan, and D. Roth. Learning to Detect Objects in Images via a Sparse, Part-Based Representation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 1475–1490, November 2004.

[4] P. Anandan. *Measuring Visual Motion from Image Sequences*. PhD thesis, University of Massachusetts, Amherst, 1987.

[5] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. BoostMap: A Method for Efficient Approximate Similarity Rankings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, D.C., July 2004.

[6] D. Avis. A Survey of Heuristics for the Weighted Matching Problem. *Networks*, 13:475–493, 1983.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[8] A. Berg. *Shape Matching and Object Recognition*. PhD thesis, U.C. Berkeley, Computer Science Division, Berkeley, CA, December 2005.

[9] A. Berg, T. Berg, and J. Malik. Shape Matching and Object Recognition using Low Distortion Correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[10] A. Berg and J. Malik. Geometric Blur for Template Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Lihue, HI, December 2001.

[11] I. Biederman. Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 44(2):115–147, 1987.

[12] S. Boughhorbel, J.-P. Tarel, and F. Fleuret. Non-Mercer Kernels for SVM Object Recognition. In *British Machine Vision Conference*, London, U.K., September 2004.

[13] S. Boughorbel, J. Tarel, and N. Boujemaa. The Intermediate Matching Kernel for Image Local Features. In *International Joint Conference on Neural Networks*, Montreal, Canada, August 2005.

[14] M. Burl, M. Weber, and P. Perona. A Probabilistic Approach to Object Recognition Using Local Photometry and Global Geometry. In *Proceedings of European Conference on Computer Vision*, Freiburg, Germany, June 1998.

[15] P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983.

[16] S. Carlsson. Order Structure, Correspondence and Shape Based Categories. In *International Workshop on Shape Contour and Grouping*, Sicily, May 1998.

[17] C. Chang and C. Lin. *LIBSVM: a library for SVMs*, 2001.

[18] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for Histogram-Based Image Classification. *Transactions on Neural Networks*, 10(5), September 1999.

[19] M. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 2002.

[20] H. Chui and A. Rangarajan. A New Algorithm for Non-Rigid Point Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, June 2000.

[21] C. L. E. Co. Poser 5 : The Ultimate 3D Character Solution. 2002.

[22] S. Cohen and L. Guibas. The Earth Mover's Distance under Transformation Sets. In *Proceedings of the IEEE International Conference on Computer Vision*, Corfu, Greece, September 1999.

[23] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[24] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial Priors for Part-Based Recognition using Statistical Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[25] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent Semantic Kernels. *Journal of Intelligent Information Systems*, 18(2/3):127–152, 2002.

[26] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual Categorization with Bags of Keypoints. In *Proceedings of European Conference on Computer Vision*, Prague, Czech Republic, May 2004.

[27] M. Cuturi and J.-P. Vert. Semigroup Kernels on Finite Sets. In *Neural Information Processing Systems*, Vancouver, Canada, December 2005.

[28] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[29] D. Demirdjian, L. Taycher, G. Shakhnarovich, K. Grauman, and T. Darrell. Avoiding the "Streetlight Effect": Tracking by Exploring Likelihood Modes. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[30] J. Eichhorn and O. Chapelle. Object Categorization with SVM: Kernels for Local Features. Technical report, MPI for Biological Cybernetics, 2004.

[31] L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: an Incremental Bayesian Approach Tested on 101 Object Cateories. In *Workshop on Generative Model Based Vision*, Washington, D.C., June 2004.

[32] P. Felzenszwalb and D. Huttenlocher. Efficient Matching of Pictorial Structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, June 2000.

[33] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning Object Categories from Google's Image Search. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[34] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

[35] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*, chapter 18. Prentice Hill, Upper Saddle River, New Jersey, 2003.

[36] T. Gartner. A Survery of Kernels for Structured Data. *Multi Relational Data Mining*, 5(1):49–58, July 2003.

[37] P. Giannopoulos and R. Veltkamp. A Pseudo-metric for Weighted Point Sets. In *Proceedings of European Conference on Computer Vision*, Copenhagen, May 2002.

[38] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, 1999.

[39] M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, November 1995.

[40] S. Gold and A. Rangarajan. A Graduated Assignment Algorithm for Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, April 1996.

[41] K. Grauman and T. Darrell. Fast Contour Matching Using Approximate Earth Mover's Distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington D.C., June 2004.

[42] K. Grauman and T. Darrell. Efficient Image Matching with Distributions of Local Invariant Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[43] K. Grauman and T. Darrell. *Nearest Neighbors in Learning and Vision: Theory and Practice*, chapter Contour Matching Using Approximate Earth Movers Distance, pages 181–202. MIT Press, 2005.

[44] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[45] K. Grauman and T. Darrell. Approximate Correspondences in High Dimensions. Technical Report MIT-CSAIL-TR-2006-045, MIT CSAIL, Cambridge, MA, June 2006.

[46] K. Grauman and T. Darrell. Pyramid Match Kernels: Discriminative Classification with Sets of Image Features. Technical Report MIT-CSAIL-TR-2006-020, MIT CSAIL, Cambridge, MA, March 2006.

[47] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3D Structure with a Statistical Image-Based Shape Model. In *Proceedings of the IEEE International Conference on Computer Vision*, Nice, France, October 2003.

[48] K. Grauman, G. Shakhnarovich, and T. Darrell. Virtual Visual Hulls: Example-Based 3D Shape Inference from Silhouettes. In *Proceedings of the 2nd Workshop on Statistical Methods in Video Processing, in conjunction with ECCV*, Prague, Czech Republic, May 2004.

[49] H. Greenspan, G. Dvir, and Y. Rubner. Region Correspondence for Image Matching via EMD Flow. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, June 2000.

[50] E. Hadjidemetriou, M. Grossberg, and S. Nayar. Multiresolution Histograms and Their Use for Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):831–847, July 2004.

[51] T. Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of Uncertainty in Artificial Intelligence*, Stockholm, 1999.

[52] T. Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42(1):177–196, January 2001.

[53] A. Holub, M. Welling, and P. Perona. Combining Generative Models and Fisher Kernels for Object Class Recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[54] A. Holub, M. Welling, and P. Perona. Exploiting Unlabelled Data for Hybrid Object Classification. Technical report, NIPS 2005 Workshop on Inter-Class Transfer, Vancouver, Canada, December 2005.

[55] P. Indyk. *High-Dimensional Computational Geometry*. PhD thesis, Stanford University, 2000.

[56] P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *30th Symposium on Theory of Computing*, 1998.

[57] P. Indyk and N. Thaper. Fast Image Retrieval via Embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision*, Nice, France, October 2003.

[58] T. Jaakkola and D. Haussler. Exploiting Generative Models in Discriminative Classifiers. In *Neural Information Processing Systems*, Denver, CO, December 1999.

[59] A. Johnson and M. Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[60] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[61] T. Kadir and M. Brady. Scale, Saliency and Image Description. *International Journal of Computer Vision*, 45(2):83–105, November 2001.

[62] S. Kamvar, D. Klein, and C. Manning. Spectral Learning. In *Proceedings of the International Conference on Artificial Intelligence*, Acapulco, Mexico, August 2003.

[63] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, D.C., June 2004.

[64] Y. Keselman, A. Shokoufandeh, M. F. Demirci, and S. Dickinson. Many-to-Many Graph Matching via Metric Embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

[65] R. Kondor and T. Jebara. A Kernel Between Sets of Vectors. In *Proccedings of International Conference on Machine Learning*, Washington, D.C., August 2003.

[66] J. Lafferty and G. Lebanon. Information Diffusion Kernels. In *Neural Information Processing Systems*, Vancouver, Canada, December 2002.

[67] S. Lazebnik, C. Schmid, and J. Ponce. A Sparse Texture Representation Using Affine-Invariant Regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

[68] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, NY, June 2006.

[69] B. Leibe and B. Schiele. Analyzing Appearance and Contour Based Methods for Object Categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

[70] T. Leung and J. Malik. Recognizing Surfaces Using Three-Dimensional Textons. In *Proceedings of the IEEE International Conference on Computer Vision*, Corfu, Greece, September 1999.

[71] E. Levina and P. Bickel. The Earth Mover's Distance is the Mallows Distance: Some Insights from Statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[72] D. Liu and T. Chen. Soft Shape Context for Iterative Closest Point Registration. In *International Conference on Image Processing*, Singapore, October 2004.

[73] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, January 2004.

[74] S. Lyu. Mercer Kernels for Object Recognition with Local Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[75] J. Malik and P. Perona. Preattentive Texture Discrimination with Early Vision Mechanisms. *Journal of Optical Society of America*, 7(5):923–932, May 1990.

[76] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, Cardiff, U.K., September 2002.

[77] K. Mikolajczyk and C. Schmid. Indexing Based on Scale Invariant Interest Points. In *Proceedings of the IEEE International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[78] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.

[79] K. Mikolajczyk and C. Schmid. Scale and Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 1(60):63–86, October 2004.

[80] MNIST handwritten digits data set. *http://yann.lecun.com/exdb/mnist*.

[81] P. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications. In *Neural Information Processing Systems*, Vancouver, December 2003.

[82] H. Muller, W. Muller, S. Marchand-Maillet, and T. Pun. Performance Evaluation in Content-Based Image Retrieval: Overview and Proposals. *Pattern Recognition Letters*, 22(5):593–601, 2001.

[83] J. Mutch and D. Lowe. Multiclass Object Recognition Using Sparse, Localized Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, NY, June 2006.

[84] NIPS conference papers. *http://www.cs.toronto.edu/ roweis/data.html*.

[85] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, NY, June 2006.

[86] F. Odone, A. Barla, and A. Verri. Building Kernels from Binary Strings for Image Matching. *IEEE Trans. on Image Processing*, 14(2):169–180, February 2005.

[87] M. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[88] A. Quattoni, M. Collins, and T. Darrel. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17*. Cambridge, MA, 2005.

[89] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. VanGool. Modeling Scenes with Local Descriptors and Latent Aspects. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[90] X. Ren and J. Malik. Learning a Classification Model for Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, Nice, France, October 2003.

[91] D. Roobaert and M. V. Hulle. View-Based 3D Object Recognition with Support Vector Machines. In *IEEE International Workshop on Neural Networks for Signal Processing*, Madison, WI, August 1999.

[92] Y. Rubner, C. Tomasi, and L. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[93] B. Scholkopf, C. Burges, and A. Smola. *Advances in Kernel Methods: Support Vector Learning*, chapter 1. The MIT Press, Cambridge, MA, 1999.

[94] T. Serre, L. Wolf, and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[95] F. Shaffalitzky and A. Zisserman. Automated Scene Matching in Movies. In *Proceedings, Challenge of Image and Video Retrieval*, London, U.K., July 2002.

[96] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, MIT, Cambridge, MA, September 2005.

[97] G. Shakhnarovich, P. Viola, and T. Darrell. Fast Pose Estimation with Parameter-Sensitive hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, Nice, France, October 2003.

[98] A. Shashua and T. Hazan. Algebraic Set Kernels with Application to Inference Over Local Image Representations. In *Neural Information Processing Systems*, Vancouver, Canada, December 2005.

[99] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[100] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[101] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering Object Categories in Image Collections. In *Proceedings of the IEEE International Conference on Computer Vision*, Beijing, China, October 2005.

[102] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*, Nice, France, October 2003.

[103] M. Stevens, B. Culbertson, and T. Malzbender. A Histogram-Based Color Consistency Test for Voxel Coloring. In *Proceedings Int. Conf. on Pattern Recognition*, Quebec, Canada, August 2002.

[104] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[105] T. Tuytelaars and L. V. Gool. Content-based Image Retrieval based on Local Affinely Invariant Regions. In *3rd Intl Conference on Visual Information Systems*, Amsterdam, the Netherlands, June 1999.

[106] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[107] R. Veltkamp and M. Hagedoorn. State-of-the-Art in Shape Matching. In *Tech Report UU-CS-1999-27*, Utrecht University, 1999.

[108] C. Wallraven, B. Caputo, and A. Graf. Recognition with Local Features: the Kernel Recipe. In *Proceedings of the IEEE International Conference on Computer Vision*, Nice, France, October 2003.

[109] G. Wang, Y. Zhang, and L. Fei-Fei. Using Dependent Regions for Object Categorization in a Generative Framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, NY, June 2006.

[110] M. Weber, M. Welling, and P. Perona. Towards Automatic Discovery of Object Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, June 2000.

[111] M. Weber, M. Welling, and P. Perona. Unsupervised Learning of Models for Recognition. In *Proceedings of European Conference on Computer Vision*, Dublin, Ireland, June 2000.

[112] J. Weston, B. Scholkopf, E. Eskin, C. Leslie, and W. Noble. Dealing with Large Diagonals in Kernel Matrices. In *Principles of Data Mining and Knowledge Discovery*, volume 243 of *SLNCS*, 2002.

[113] C. Williams and M. Seeger. Using the Nystrom Method to Speed Up Kernel Machines. In *Neural Information Processing Systems*, Vancouver, Canada, 2001.

[114] L. Wolf, S. Bileschi, and E. Meyers. Perception Strategies in Hierarchical Vision Systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, NY, June 2006.

[115] L. Wolf and A. Shashua. Learning Over Sets Using Kernel Principal Angles. *Journal of Machine Learning Research*, 4:913–931, December 2003.

[116] WordNet. *http://wordnet.princeton.edu*.

[117] T. Yeh, K. Grauman, K. Tollmar, and T. Darrell. A Picture is Worth a Thousand Keywords: Image-Based Object Search on a Mobile Platform. In *Conference on Human Factors in Computing Systems (CHI)*, Portland, OR, April 2005.

[118] L. Zelnik-Manor and P. Perona. Self-Tuning Spectral Clustering. In *Neural Information Processing Systems*, Vancouver, Canada, December 2004.

[119] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, NY, June 2006.

[120] H. Zhang and J. Malik. Learning a Discriminative Classifier Using Shape Context Distances. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.